Security and Safety, Vol. 1, 2021001 (2022)
https://doi.org/10.1051/sands/2021001
sands.edpsciences.org

Review OPEN ACCESS

Information Network

Concretely efficient secure multi-party computation protocols: survey and more

Dengguo Feng[1,2] and Kang Yang[1,*]

[1] *State Key Laboratory of Cryptology, Beijing 100878, China*
[2] *State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China*

Received: 11 October 2021 / Revised: 5 November 2021 / Accepted: 6 December 2021 / Published online: 14 June 2022

Abstract Secure multi-party computation (MPC) allows a set of parties to jointly compute a function on their private inputs, and reveals nothing but the output of the function. In the last decade, MPC has rapidly moved from a purely theoretical study to an object of practical interest, with a growing interest in practical applications such as privacy-preserving machine learning (PPML). In this paper, we comprehensively survey existing work on concretely efficient MPC protocols with both semi-honest and malicious security, in both dishonest-majority and honest-majority settings. We focus on considering the notion of security with abort, meaning that corrupted parties could prevent honest parties from receiving output after they receive output. We present high-level ideas of the basic and key approaches for designing different styles of MPC protocols and the crucial building blocks of MPC. For MPC applications, we compare the known PPML protocols built on MPC, and describe the efficiency of private inference and training for the state-of-the-art PPML protocols. Furthermore, we summarize several challenges and open problems to break though the efficiency of MPC protocols as well as some interesting future work that is worth being addressed. This survey aims to provide the recent development and key approaches of MPC to researchers, who are interested in knowing, improving, and applying concretely efficient MPC protocols.

Keywords Secure multi-party computation, Privacy-preserving machine learning, Secret sharings, Garbled circuits, Oblivious transfer and its arithmetic generalization

Citation Feng D and Yang K. Concretely efficient secure multi-party computation protocols: survey and more. Security and Safety 2022; **1**: 2021001. https://doi.org/10.1051/sands/2021001

1 Introduction

Secure multi-party computation (MPC) allows a set of parties to jointly compute a function on their private inputs without revealing anything but the output of the function. Specifically, MPC allows n parties to jointly compute the following function:

$$(y_1, \ldots, y_n) \leftarrow f(x_1, \ldots, x_n),$$

where every party P_i holds an input x_i, obtains an output y_i, and can learn nothing except for (x_i, y_i, f), and function f is often modeled as a Boolean or arithmetic circuit. MPC is a foundation of cryptography, and is also a core technology to protect privacy of data for cooperative computing in the big data era.

* Corresponding author (email: yangk@sklc.org)

In the late 1980s, it was shown that MPC protocols for any function can be constructed [1–6], which demonstrates the feasibility of MPC. However, significant improvements are necessary to make MPC sufficiently efficient to be used in practice. In the last decade, MPC has been developed from a theoretical field to a practical one, and is starting to see real-life deployments. A series of open-sourced libraries for MPC (*e.g.*, ABY [7], EMP-toolkit [8], FRESCO [9], JIFF [10], MP-SPDZ [11], MPyC [12], SCALE-MAMBA [13], Bogdanov *et al.* [14], and TinyGable [15]) have been developed, and further promote the applications and deployment of MPC. We refer the reader to [16, 17] for more MPC libraries and the comparison of known libraries. Many applications can be built upon MPC to protect the privacy of data, including machine learning (see, *e.g.*, [18–31] and references therein), federated learning (see, *e.g.*, [32–36]), data mining [37–40], auction [41–43], genomic analysis [44–46], securing databases (see [47] and references therein), and blockchain [48–53]. In addition, some techniques underlying MPC protocols can also be used to construct non-interactive zero-knowledge (ZK) proofs [54–61] based on the MPC-in-the-head paradigm, scalable ZK proofs [62–71], threshold cryptography (see [72–79] and references therein), and private set intersection (PSI) (see, *e.g.*, [80–85]).

Secure multi-party computation guarantees privacy (meaning that the protocol reveals nothing but the output) and correctness (meaning that the correct function is computed), among others (*e.g.*, independence of inputs, meaning that a party cannot choose its input as a function of the other parties' inputs). The security properties must be guaranteed in the presence of adversarial behaviors. We mainly consider two classic adversaries:

- **Semi-honest:** Semi-honest adversaries (a.k.a., passive adversaries) follow the protocol specification but may try to learn more than allowed from the protocol transcript;
- **Malicious:** Malicious adversaries (a.k.a., active adversaries) can run an arbitrary attack strategy in its attempt to break the protocol.

According to the maximum number t of corrupted parties, there are typically two settings considered in the literature: **dishonest majority** ($n/2 \leq t < n$, particularly we often adopt $t = n - 1$) and **honest majority** ($t < n/2$), where n is the total number of parties. In addition to the basic properties (privacy and correctness), some applications might further require fairness meaning that if one party obtains output then so do all, or guaranteed output delivery (GOD) meaning that all parties always receive output. To obtain better efficiency, we often consider a relaxed security notion, called security with abort, which allows corrupted parties to prevent honest parties from receiving output after the corrupted parties received their output. This security notion is standard in the dishonest-majority setting, as not all functions can be computed fairly without an honest majority [86]. We focus on considering the notion of security with abort and the static adversary that determines the set of corrupted parties at the onset of protocol, which allows to achieve the best efficiency. In this paper, we mainly consider concretely efficient MPC protocols, and ignore the MPC protocols that have a good asymptotic complexity or optimal round complexity, but their concrete efficiency is very low.

There are two main approaches for designing MPC protocols [87]: (1) the secret-sharing approach (followed by [2–4]) that makes the parties interact for every non-linear gate of the circuit, and has a low communication bandwidth but a number of rounds linear to the circuit depth; (2) the garbled-circuit approach (followed by [1, 6]) that lets the parties construct an encrypted version of the circuit allowing to be computed only once, and has a constant number of rounds but a large communication bandwidth. In general, the secret-sharing approach is more suitable for low-latency networks such as local area network (LAN), while the garbled-circuit approach will perform better in high-latency networks such as wide area network (WAN).

There are a lot of concretely efficient MPC protocols that have been proposed, and these MPC protocols give a different trade-off in terms of security and efficiency. Thus, it is not easy to make the best choice of MPC protocols in concrete applications for non-experts. In addition, many techniques have been developed for MPC, which makes it difficult for new researchers to understand the techniques in a short period. Through this survey, we aim to help the researchers, who are interested in MPC, rapidly understand the recent development and some basic/key ideas of concretely efficient MPC protocols, and be able to make a better choice in terms of applying MPC protocols.

1.1 Our contribution

In this paper, we comprehensively survey the known work on concretely efficient MPC protocols with both semi-honest and malicious security. This survey not only covers the work in the dishonest-majority setting, but also concerns on the recent development in the honest-majority setting. Our survey involves not only secret-sharing based MPC but also garbled-circuit based MPC. We present the basic approaches for designing concretely efficient MPC protocols, and give the high-level ideas for the key techniques underlying these MPC protocols. In particular, we give a uniform framework and view of MPC protocols based on additive, Shamir, and replicated secret sharings. We also provide a high-level view of the recent approach with sublinear communication to design correlated oblivious transfer (COT) that is a crucial building block for dishonest-majority MPC. In addition, we describe one important application of MPC (*i.e.*, privacy-preserving machine learning, or PPML in short), and summarize the known PPML protocols in terms of functionality, security, techniques and neural-network architectures as well as discussing several key techniques for designing concretely efficient PPML protocols. Furthermore, we propose several challenges and interesting open problems to break through the efficiency of differently flavored MPC protocols, and also present some future work that need to be addressed for developing or deploying MPC.

Comparison with other MPC surveys. Recently, Lindell [88] presented a short MPC survey, which gives an overview of the security of MPC, a summarization of MPC feasible results, an honest-majority MPC framework based on Shamir secret sharing, two specific MPC protocols (*i.e.*, PSI and threshold RSA), a very short overview of dishonest-majority MPC, and several application examples of MPC. This survey focuses on the notion and security of MPC and how MPC is being currently used, and does not involve the recent development of MPC protocols which is addressed by our survey. Besides, Orsini [89] gave a survey for only SPDZ-style MPC protocols in the dishonest-majority malicious setting. Compared to the two surveys [88, 89], our survey is more comprehensive, presents an important MPC application (*i.e.*, PPML) and also gives interesting future work for MPC.

1.2 Organization

In Section 2, we define the necessary notation to be used in the subsequent main body, and then describe the security and communication models for MPC and give a sketch of several basic building blocks of MPC. We describe the development and the key techniques for MPC based on secret sharings (resp., garbled circuits) in Section 3 (resp., Section 4). We present the crucial building blocks for dishonest-majority MPC in Section 5. In Section 6, we summarize the protocols that apply MPC to realize private inference and training of machine learning, as well as several key techniques for PPML applications. Finally, we conclude this survey, and propose open problems and future work for MPC.

2 Preliminaries

2.1 Notation

We use κ and ρ to denote the computational and statistical security parameters, respectively. In the known MPC implementations, we often adopt $\kappa = 128$ and $\rho = 40$, where sometimes we also consider $\rho = 64$ or $\rho = 80$. For two integers a, b with $a < b$, we denote by $[a, b]$ the set $\{a, \ldots, b\}$ and by $[a, b)$ the set $\{a, \ldots, b-1\}$. We use $x \leftarrow S$ to denote sampling x uniformly at random from set S and $x \leftarrow \mathcal{D}$ to denote sampling x according to distribution $\mathcal{D}$. For bit-string x, we use $\mathsf{lsb}(x)$ to denote the least significant bit of x. For row vector $\mathbf{x}$, we use x_i to denote the ith component of $\mathbf{x}$ with x_1 the first entry, and denote by $\mathsf{HW}(\mathbf{x})$ the Hamming weight of $\mathbf{x}$ (*i.e.*, the number of non-zero entries in vector $\mathbf{x}$). For two families of distributions $X = \{X_\kappa\}_{\kappa\in\mathbb{N}}$ and $Y = \{Y_\kappa\}_{\kappa\in\mathbb{N}}$, we write $X \overset{\mathrm{c}}{\approx} Y$ if X and Y are computationally indistinguishable.

We use $\mathbb{F}$ to denote a finite field defined by a prime or a power of prime, and denote by $|\mathbb{F}|$ the size of field $\mathbb{F}$. In particular, we often consider a binary field $\mathbb{F} = \mathbb{F}_2$ and an arithmetic field $\mathbb{F} = \mathbb{F}_p$ for a large prime p. Let $\mathbb{K}$ be a field extension of $\mathbb{F}$ of degree k. We fix some monic, irreducible polynomial $f(X)$ of degree k and write $\mathbb{K} \cong \mathbb{F}[X]/f(X)$. Then, we can denote every element $y \in \mathbb{K}$ by a polynomial $y_0 + y_1 \cdot X + \cdots + y_{k-1} \cdot X^{k-1}$ over $\mathbb{F}$, and view y as a vector $(y_0, y_1, \ldots, y_{k-1}) \in \mathbb{F}^k$. When we write

arithmetic expressions involving both elements of $\mathbb{F}$ and elements of $\mathbb{K}$, it means that values in $\mathbb{F}$ are viewed as polynomials lying in $\mathbb{K}$ with only constant terms. Specifically, we use $\mathbb{F}_{2^\kappa}$ to denote the degree-κ extension field of $\mathbb{F}_2$. In the context of MPC for Boolean circuits, we often use $\{0,1\}^\kappa$, $\mathbb{F}_2^\kappa$ and $\mathbb{F}_{2^\kappa}$ interchangeably, and thus addition in $\mathbb{F}_2^\kappa$ or $\mathbb{F}_{2^\kappa}$ corresponds to XOR in $\{0,1\}^\kappa$.

Most of the known MPC protocols model a function as a circuit. Specifically, circuit $\mathcal{C}$ over an arbitrary field $\mathbb{F}$ is defined by a set of input wires and output wires along with a list of gates of the form $(\alpha, \beta, \gamma, T)$, where α, β are the indices of the input wires of the gate, γ is the index of the output wire of the gate, and $T \in \{\mathsf{ADD}, \mathsf{MULT}\}$ is the type of the gate. If $\mathbb{F} = \mathbb{F}_2$, then $\mathcal{C}$ is a Boolean circuit with $\mathsf{ADD} = \oplus$ and $\mathsf{MULT} = \wedge$. Otherwise, $\mathcal{C}$ is an arithmetic circuit where $\mathsf{ADD}/\mathsf{MULT}$ corresponds to addition/multiplication in field $\mathbb{F}$. We let $|\mathcal{C}|$ denote the number of MULT gates in the circuit.

We will use n and t to denote the number of total parties and the number of corrupted parties respectively, unless otherwise specified. Sometimes, we call t as the corruption threshold.

2.2 Security and communication models

Security model. All security properties for MPC can be formalized in an Ideal/Real paradigm [90, 91], which provides a modular way to prove security. The real-world execution where the parties interact with semi-honest/malicious adversary $\mathcal{A}$ and execute a protocol Π is compared to the ideal-world execution where the parties interact with a simulator $\mathcal{S}$ and an ideal functionality $\mathcal{F}$. In the ideal word, $\mathcal{F}$ plays the role of an incorruptible trusted party, and captures the security of MPC protocols. We use $P_1, \ldots, P_n$ to denote n parties running a protocol. In the multi-party setting (*i.e.*, $n > 2$), we often consider a rushing adversary $\mathcal{A}$, meaning that $\mathcal{A}$ is allowed to receive its incoming messages in a round before it sends its outgoing message. If the adversary is allowed to be computationally unbounded, the protocol is said to be information-theoretically secure. If the adversary is bounded to (non-uniform) probabilistic polynomial time (PPT), the protocol satisfies the computational security.

The known MPC protocols mainly use two types of simulation models: stand-alone setting [91, 92] and universal composability (UC) [90]. Compared to stand-alone model, UC model additionally involves an environment $\mathcal{Z}$ which can determine the inputs of honest parties and receive the outputs from the honest parties. This may make security proofs of MPC protocols somewhat more complex. While stand-alone model only guarantees the security under the sequential composition, UC model has the property that the protocols maintain their security, even though running concurrently with other (in)secure protocols. According to the result from [93], we obtain that any MPC protocol in the stand-alone model, which is proven secure with a black-box straight-line simulator and has the property the inputs of all parties are fixed before the protocol execution begins (referred as to start synchronization or input availability), is also secure under concurrent composition.

Communication model. The default method of communication for MPC is authenticated channel, which can be implemented in practice using the known standard techniques. In the multi-party setting, all parties are also connected *via* point-to-point channels, and sometimes need also a broadcast channel. A broadcast channel can be efficiently implemented using a standard 2-round echo-broadcast protocol [94], as we only consider security with abort. The communication overhead of this broadcast protocol can be significantly improved by letting all parties send the hash outputs of received messages and aborting if an inconsistent hash value is detected. Sometimes, the parties need to communicate over a private channel, meaning that the messages sent over such channel are kept confidential and authenticated. As such, private channel can be established using the standard techniques.

2.3 Oblivious transfer and oblivious linear-function evaluation

Informally, oblivious transfer (OT) [95, 96] involves two parties where a sender inputs two messages (m_0, m_1) and a receiver inputs a choice bit b, and allows the receiver to obtain m_b while keeping b secret against the sender and m_{1-b} confidential against the receiver. We use an OT extension protocol to generate a large number of OT correlations (see Section 5). Oblivious linear-function evaluation (OLE) is an arithmetic generalization of OT to a large field $\mathbb{F}$, and is a special case of oblivious polynomial

evaluation introduced in [97]. Specifically, OLE is a two-party protocol between a sender and a receiver, where the sender inputs $u, v \in \mathbb{F}$, and the receiver inputs $x \in \mathbb{F}$ and obtains an output $y = u \cdot x + v \in \mathbb{F}$. We discuss the development of OT/OLE and their variants in Section 5.

2.4 Commitment and coin-tossing

Commitment. In the dishonest-majority setting, MPC protocols often need a commitment scheme to commit a value while keeping it secret (the hiding property), and then to open the value while keeping it consistent with the one that has been committed (the binding property). To achieve high efficiency, the commitment scheme is often constructed in the random-oracle model by defining $\mathsf{Commit}(x) = \mathsf{H}(x, r)$, where x is a message, r is a randomness, and $\mathsf{H} : \{0,1\}^* \to \{0,1\}^{2\kappa}$ is a cryptographic hash function modeled as a random oracle.

Coin-tossing. A lot of MPC protocols with malicious security require the parties to generate multiple public randomness by executing a coin-tossing protocol, while guaranteeing that no malicious parties can control the randomness or make them deviate from uniform distribution. In the dishonest-majority setting, the coin-tossing protocol is constructed by that (1) every party P_i commits a random seed s_i and then opens the seed; (2) every party computes $s := \bigoplus_{i\in[n]} s_i$ and then generates the public randomness with s and a pseudorandom generator (PRG) modeled as a random oracle. In the honest-majority setting, the coin-tossing protocol is constructed in a totally different way. Specifically, all parties generate multiple random shares, and then open these shares as the public randomness, where random shares can be generated very efficiently when a majority of parties are honest.

3 MPC protocols based on secret sharings

Based on the secret-sharing approach, the concretely efficient MPC protocols enable the parties to send small messages per non-linear gate, but has a number of rounds linear to the depth of the circuit being computed. For now, concretely efficient MPC protocols mainly adopt three kinds of linear secret-sharing schemes (LSSSs): additive secret sharing, Shamir secret sharing [98], and replicated secret sharing (a.k.a., CNF secret sharing) [99, 100], where additive secret sharing is mainly used for MPC protocols in the dishonest-majority setting, while Shamir and replicated secret sharings are adopted for honest-majority MPC protocols. We first recall the constructions of these LSSSs in a uniform view. To achieve malicious security, additive secret sharing needs to be equipped with information-theoretic message authentication codes (IT-MACs), and thus we define two types of IT-MACs used in MPC with dishonest majority. Note that IT-MACs are unnecessary for Shamir/replicated secret sharing in the honest-majority setting. Then, based on LSSSs, we describe how to construct semi-honest MPC protocols with a uniform structure. Finally, we present how to transform semi-honest MPC protocols to maliciously secure MPC protocols using the state-of-the-art checking techniques.

3.1 Linear secret-sharing schemes

All the three kinds of LSSSs used in MPC are (n, t)-threshold secret sharing schemes, which enable n parties to share secret x among the parties, such that no subset of t parties can obtain any information on secret x, while any subset of $t + 1$ parties can reconstruct secret x. Additive secret sharing can only be made for $t = n - 1$, while Shamir/replicated secret sharing allows any $t < n$ (we often adopt $t < n/2$ for honest-majority MPC). The three kinds of LSSSs are defined over a field $\mathbb{F}$. While additive/replicated secret sharing allows an any-sized field (including $\mathbb{F}_2$), Shamir secret sharing requires $|\mathbb{F}| > n$. Below, we describe the constructions of these LSSSs and the useful procedures for designing MPC protocols.

- $\mathsf{Share}(x)$: For $x \in \mathbb{F}$, a dealer generates n shares $x^1, \ldots, x^n$. By $[x]$, we denote the sharing of x.
  (a) *Additive secret sharing:* This is the simplest LSSS as far as we know. To share a value $x \in \mathbb{F}$, the dealer samples $x^i \leftarrow \mathbb{F}$ for $i \in [1, n]$ such that $\sum_{i\in[1,n]} x^i = x$, and sends x^i to P_i.

(b) *Shamir secret sharing:* Let $\alpha_1, \ldots, \alpha_n \in \mathbb{F}$ be n distinct non-zero elements (*e.g.*, $\alpha_i = i$ for $i \in [1, n]$). The dealer samples a random polynomial $f(X)$ of degree t over $\mathbb{F}$ such that $f(0) = x$, and then sends $x^i = f(\alpha_i)$ to P_i. Shamir secret sharing is mainly used for a large n in the honest-majority MPC protocols.

(c) *Replicated secret sharing:* The dealer samples $x^T \leftarrow \mathbb{F}$ for $T \in \mathcal{T}$ such that $\sum_{T \in \mathcal{T}} x^T = x$, where $\mathcal{T}$ consists of all sets of t parties. Every party P_i obtains shares x^T for all $T \in \mathcal{T}$ subject to $i \notin T$. In general, the total number of shares is $\binom{n}{t}$ and every party stores $\binom{n-1}{t}$ shares, which can become very large as n and t grow. Therefore, we mainly use replicated secret sharing when n is small. For example, if $n = 3$ and $t = 1$, then the dealer samples $x^1, x^2, x^3 \leftarrow \mathbb{F}$ such that $x^1 + x^2 + x^3 = x$, and then sends (x^2, x^3) to P_1, (x^1, x^3) to P_2 and (x^1, x^2) to P_3.

Note that the shares for replicated/Shamir secret sharing need to be sent over a private channel, but this may be not necessary for additive secret sharing in most cases. In MPC protocols, either a party P_i acts as a dealer if it knows secret x, or the computation of a dealer is performed distributedly by a protocol if no one knows x. In the computational setting, we can use the pseudo-random secret sharing (PRSS) approach [99] to reduce the communication cost that distributes the shares of Shamir and replicated sharings. For example, to generate a degree-t Shamir sharing $[x]_t$, the dealer can send a random seed s_i to P_i who computes x^i with s_i and a pseudo-random generator PRG for each $i \in [1, t]$. Then, the dealer sends $x^{t+1}, \ldots, x^n$ to $P_{t+1}, \ldots, P_n$, respectively, such that $(x^1, \ldots, x^{t+1})$ defines a degree-t polynomial f with $f(0) = x$ and $x^i = f(\alpha_i)$ for $i \in [t + 2, n]$. Since the seeds $\mathsf{s}_1, \ldots, \mathsf{s}_t$ can be reused for multiple Shamir sharings, the communication to send the seeds could be ignored. Besides, based on the PRSS technique [99], we can convert a random replicated sharing into a random Shamir sharing with a small communication to distribute the seeds. This technique is only suitable for a small number of parties, as the number of random seeds is $\binom{n}{t}$ and grows exponentially with the number of parties.

- $\mathsf{Reconstruct}([x], i)$: This procedure enables only P_i to obtain secret x. When any $t + 1$ shares of $[x]$ are sent to P_i over a private channel, secret x can be reconstructed by P_i as follows:
  (a) *Additive secret sharing:* Given $\{x^j\}_{j \neq i}$ from all other parties, P_i computes $x := \sum_{i \in [1,n]} x^i$.
  (b) *Shamir secret sharing:* Secret x can be reconstructed using Lagrange interpolation. Without loss of generality, we assume that P_i gets shares $x^1, \ldots, x^{t+1}$. Then P_i can compute $x := \sum_{i \in [1,t+1]} \delta_i(0) \cdot x^i$, where $\delta_i(X) = \prod_{j \in [1,t+1], j \neq i} (X - \alpha_j)/(\alpha_i - \alpha_j)$ is a degree-t polynomial.
  (c) *Replicated secret sharing:* Given x^T for all $T \in \mathcal{T}$ with $i \in T$, P_i computes $x := \sum_{T \in \mathcal{T}} x^T$. Specifically, for $(n, t) = (3, 1)$, after receiving x^i from one party, P_i computes $x := x^1 + x^2 + x^3$.
- $\mathsf{Open}([x])$: This procedure allows all parties to know x. This is easy to be realized by executing $\mathsf{Reconstruct}([x], i)$ for $i \in [1, n]$, where a private channel is unnecessary.
- **Linear combination:** Given public coefficients $c_1, \ldots, c_\ell, c \in \mathbb{F}$, all parties can compute locally $[y] := \sum_{h \in [1,\ell]} c_h \cdot [x_h] + c$ such that $y = \sum_{h \in [1,\ell]} c_h \cdot x_h + c$ as follows:
  (a) *Additive secret sharing:* For $i \neq 1$, P_i computes $y^i := \sum_{h \in [1,\ell]} c_h \cdot x_h^i$ where x_h^i is P_i's share of x_h. Party P_1 computes $y^1 := \sum_{h \in [1,\ell]} c_h \cdot x_h^1 + c$.
  (b) *Shamir secret sharing:* For $i \in [1, n]$, P_i computes $y^i := \sum_{h \in [1,\ell]} c_h \cdot x_h^i + c$.
  (c) *Replicated secret sharing:* Fix a set $T_1 \in \mathcal{T}$. For $i \in T_1$, P_i computes $y^T := \sum_{h \in [1,\ell]} c_h \cdot x_h^T + c$ for all $T \in \mathcal{T}$ subject to $i \notin T$ where x_h^T is the P_i's share of x_h. For $i \in [1, n], i \notin T_1$, P_i computes $y^T := \sum_{h \in [1,\ell]} c_h \cdot x_h^T$ for all $T \in \mathcal{T}$ with $i \notin T$. In particular, when $n = 3$ and $t = 1$, $y^i = \sum_{h \in [1,\ell]} c_h \cdot x_h^i$ if $i \neq 1$ and $y^i = \sum_{h \in [1,\ell]} c_h \cdot x_h^i + c$ otherwise, where P_i gets $\{y^j\}_{j \neq i}$.

The LSSSs defined as above guarantee perfect privacy in the presence of malicious adversaries, but only provide correctness against semi-honest adversaries. To achieve malicious security for the case of $t < n/2$, we need to modify the $\mathsf{Reconstruct}$ procedure as follows:

- $\mathsf{Reconstruct}([x], i)$: When all shares of $[x]$ are sent to P_i over a private channel, P_i can reconstruct secret x as follows:
  (a) *Shamir secret sharing:* After receiving all shares $x^1, \ldots, x^n$ of $[x]$, P_i can compute a polynomial $f(X) := \sum_{j \in [1,t+1]} \delta_j(X) \cdot x^j$, where $\delta_j(X) = \prod_{k \in [1,t+1], k \neq j} \frac{X - \alpha_k}{\alpha_j - \alpha_k}$ is a degree-t polynomial.

Then, for $j \in [1,n]\backslash[1,t+1]$, P_i computes $f(\alpha_j)$ and checks that $x^j = f(\alpha_j)$. If the check fails, P_i aborts; otherwise, it computes $x := f(0)$. If the check passes, then all shares are computed from a unique polynomial $f(X)$. We know that $t+1$ shares of n shares are correct from honest parties, and uniquely determine the polynomial $f(X)$. Therefore, x reconstructed by P_i is correct, if P_i does not abort.

If multiple secret values need to be reconstructed, then we can optimize the communication similarly using a cryptographic hash function. In particular, P_j for all $j \in [1,n]\backslash[1,t+1]$ send the hash values of their shares to P_i, and then P_i computes their shares using polynomial interpolation and checks whether these hash values are correct.

(b) *Replicated secret sharing:* For the sake of simplicity, we focus on the case that only one party of three parties allows to be corrupted (*i.e.*, $n = 3$ and $t = 1$). For other honest-majority cases, this procedure can be done similarly.

Concretely, P_{i-1} sends $x^i_{(i-1)}$ to P_i and P_{i+1} sends $x^i_{(i+1)}$ to P_i, where the subscript indexes are computed module 3. Then P_i checks that $x^i_{(i-1)} = x^i_{(i+1)}$. If the check fails, P_i aborts; otherwise, it defines $x^i := x^i_{(i-1)}$ and computes $x := x^1 + x^2 + x^3$. Since either P_{i-1} or P_{i+1} is honest, the equality check can guarantee that share x^i is correct, and thus the secret x reconstructed by P_i is correct.

If $[x_1], \ldots, [x_\ell]$ need to be reconstructed, then we can further reduce the communication of this procedure using a cryptographic hash function $\mathsf{H} : \{0,1\}^* \to \{0,1\}^\kappa$ [101]. In particular, P_{i-1} sends $x^i_1, \ldots, x^i_\ell$ to P_i, while P_{i+1} sends $\tau := \mathsf{H}(\tilde{x}^i_1, \ldots, \tilde{x}^i_\ell)$ to P_i. Then, P_i checks $\tau = \mathsf{H}(x^i_1, \ldots, x^i_\ell)$. If the check fails, P_i aborts; otherwise, it computes $x_h := x^1_h + x^2_h + x^3_h$ for $h \in [1,\ell]$. If P_{i-1} is honest, then it is clear that P_i will obtain the correct secrets. Otherwise (*i.e.*, P_{i+1} is honest), if there exists some $h \in [1,\ell]$ such that $x^i_h \neq \tilde{x}^i_h$, then P_i will abort except with probability $\mathsf{negl}(\kappa)$, based on the second pre-image resistance of H.

For additive secret sharing, we need to equip it with IT-MACs to guarantee the security of procedures $\mathsf{Reconstruct}$ and Open in the presence of malicious adversaries, which is shown in the next subsection.

3.2 Information-theoretic message authentication codes

In the dishonest-majority setting, MPC protocols can use additive secret sharing to execute the circuit evaluation privately [4, 102]. This is sufficient for semi-honest security. Nevertheless, in the malicious setting, IT-MACs need to be introduced to guarantee the correctness of secret values [103, 104]. Currently, there are two-style IT-MACs that are used in MPC protocols: BDOZ-style [105] and SPDZ-style [103]. While the original IT-MACs are defined over a single large field, it is easy to extend them so that values are defined over an any-sized field $\mathbb{F}$ and authentication is done over a large extension field $\mathbb{K}$, which is described as follows:

- **BDOZ-style IT-MACs [105]:** Sample a global key (a.k.a. MAC key) $\Delta \leftarrow \mathbb{K}$. For a message $x \in \mathbb{F}$, sample a local key $\mathsf{K} \leftarrow \mathbb{K}$, and define an MAC on x as $\mathsf{M} = \mathsf{K} + x \cdot \Delta$, where (x, M) is held by a party $\mathsf{P_A}$ and (K, Δ) is produced by the other party $\mathsf{P_B}$.
  If a malicious $\mathsf{P_A}$ forges an MAC M' on $x' \neq x$ such that $\mathsf{M}' = \mathsf{K} + x' \cdot \Delta$, then $\mathsf{P_A}$ learns $\Delta = (\mathsf{M} - \mathsf{M}')/(x - x')$, which occurs with probability $1/|\mathbb{K}|$ as Δ is perfectly hidden.
- **SPDZ-style IT-MACs [103]:** Sample a global key $\Delta \leftarrow \mathbb{K}$. For a message $x \in \mathbb{F}$, the MAC on x is defined as $\mathsf{M} = x \cdot \Delta$. Note that every party holds the additive shares of Δ and M, and no party knows the key Δ and the MAC M (see below for details). The security analysis is similar to that of BDOZ-style IT-MACs.

It is easy to see that SPDZ-style IT-MACs are more compact than BDOZ-style IT-MACs. Nevertheless, when applying to MPC, it is incomparable for the IT-MACs of two styles. While BDOZ-style IT-MACs are more suitable to be used in constant-round MPC protocols based on distributed garbling [106–111], SPDZ-style IT-MACs are mainly adopted to transform the semi-honest GMW protocol [4] into efficient MPC protocols with malicious security.

Given the above IT-MACs, we can define authenticated secret sharing (*i.e.*, additive secret sharing with IT-MACs authentication) as follows:

- $\mathsf{Initialize}$: For each $i \in [1,n]$, a dealer samples $\Delta_i \leftarrow \mathbb{K}$, and sends Δ_i to P_i. For SPDZ-style IT-MACs, the dealer also computes $\Delta := \sum_{i\in[1,n]} \Delta_i$, where Δ_i is the P_i's share of Δ and can be also written as Δ^i in this case.
- $\mathsf{Share}(x)$: For $x \in \mathbb{F}$, the dealer generates n random additive shares $x^1, \ldots, x^n \in \mathbb{F}$ such that $\sum_{i\in[1,n]} x^i = x$, where x^i is the P_i's share. Then, the dealer computes their MAC tags as follows:
  (a) **BDOZ-style:** Each share x^i is authenticated independently by $n-1$ different parties. In particular, for each $j \neq i$, sample $\mathsf{K}_j[x^i] \leftarrow \mathbb{K}$ and compute $\mathsf{M}_j[x^i] := \mathsf{K}_j[x^i] + x^i \cdot \Delta_j$. Then send $\big(x^i, \{(\mathsf{M}_j[x^i], \mathsf{K}_i[x^j])\}_{j\neq i}\big)$ to every party P_i.
  (b) **SPDZ-style:** Compute $\mathsf{M} := x \cdot \Delta$, and then sample uniform MAC shares $\mathsf{M}^1, \ldots, \mathsf{M}^n$ such that $\sum_{i\in[1,n]} \mathsf{M}^i = x \cdot \Delta$, *i.e.*, $\sum_{i\in[1,n]} \mathsf{M}^i = (\sum_{i\in[1,n]} x^i) \cdot (\sum_{i\in[1,n]} \Delta^i)$.
  By $[\![x]\!]$, we denote the authenticated sharing of x.
- $\mathsf{Reconstruct}([\![x_1]\!], \ldots, [\![x_\ell]\!], i)$: This procedure enables only P_i to obtain secrets $x_1, \ldots, x_\ell$ for some $\ell \in \mathbb{N}$ by executing as follows:
  (a) **BDOZ-style:** Let $\mathsf{H} : \{0,1\}^* \to \{0,1\}^\kappa$ be a hash function modeled as a random oracle. For each $j \neq i$, P_j computes $\tau_j := \mathsf{H}(\mathsf{M}_i[x_1^j], \ldots, \mathsf{M}_i[x_\ell^j])$, and then sends $(x_1^j, \ldots, x_\ell^j, \tau_j)$ to P_i over a private channel. Then, for $j \neq i$, P_i checks that $\tau_j = \mathsf{H}(\mathsf{K}_i[x_1^j] + x_1^j \cdot \Delta_i, \ldots, \mathsf{K}_i[x_\ell^j] + x_\ell^j \cdot \Delta_i)$ holds. The soundness error is bounded by $(n-1)/|\mathbb{K}| + (n-1)/2^\kappa$ based on the analysis [112]. Following the work [112], we can define $\mathsf{H}(m_1, \ldots, m_\ell) = \bigoplus_{i\in[1,\ell]} F(m_i)$ and use the fixed-key AES to instantiate F, where the computation of fixed-key AES is blazing fast given hardware-instruction support.
  (b) **SPDZ-style:** Let $[\![r]\!]$ be a random authenticated share. Every party sends its shares on $[\![x_1]\!], \ldots, [\![x_\ell]\!], [\![r]\!]$ to P_i over a private channel, and then P_i can reconstruct $x_1, \ldots, x_\ell$ and r. All parties run a coin-tossing protocol to generate uniform elements $\chi_1, \ldots, \chi_\ell \in \mathbb{K}$, and compute $[\![y]\!] := \sum_{h\in[1,\ell]} \chi_h \cdot [\![x_h]\!] + [\![r]\!]$. Then P_i computes $y := \sum_{h\in[1,\ell]} \chi_h \cdot x_h + r$ and sends it to all other parties. For each $j \neq i$, P_j sends $\sigma^j := \mathsf{M}(y)^j - y \cdot \Delta^j$ to P_i. Then P_i computes $\sigma^i := \mathsf{M}(y)^i - y \cdot \Delta^i$ and checks that $\sum_{h\in[1,n]} \sigma^h = 0$.
- $\mathsf{Open}([\![x_1]\!], \ldots, [\![x_\ell]\!])$: This procedure allows all parties to get $x_1, \ldots, x_\ell$.
  (a) **BDOZ-style:** This is easy to be realized by executing $\mathsf{Reconstruct}([\![x_1]\!], \ldots, [\![x_\ell]\!], i)$ for $i \in [1,n]$, where a private channel is unnecessary.
  (b) **SPDZ-style:** Every party broadcasts its shares on $[\![x_1]\!], \ldots, [\![x_\ell]\!]$ to all other parties, and then computes the values $x_1, \ldots, x_\ell$. To verify the correctness of $x_1, \ldots, x_\ell$, all parties execute the following batch-check procedure:
    (i) All parties execute a coin-tossing protocol to sample uniformly random $\chi_1, \ldots, \chi_\ell \in \mathbb{K}$.
    (ii) The parties compute $[\![y]\!] := \sum_{h\in[1,\ell]} \chi_h \cdot [\![x_h]\!]$.
    (iii) Every party P_i computes $\sigma^i := \mathsf{M}(y)^i - y \cdot \Delta^i$, and then commit σ^i to all other parties.
    (iv) Every party P_i opens σ^i to all other parties.
    (v) Every party P_i checks that $\sum_{h\in[1,n]} \sigma^h = 0$.

For MPC protocols in the dishonest-majority setting, the dealer is realized distributedly by executing a protocol, where global key Δ_i is sampled uniformly at random by party P_i. Note that authenticated shares (*i.e.*, additive secret sharing equipped with IT-MACs) are still additively homomorphic, as IT-MACs are additively homomorphic. That is, given public coefficients $c_1, \ldots, c_\ell, c \in \mathbb{F}$, all parties can compute locally $[\![y]\!] := \sum_{h\in[1,\ell]} c_h \cdot [\![x_h]\!] + c$. Authenticated shares of both styles are able to be generated using COT or its arithmetic generalization vector oblivious linear-function evaluation (VOLE), which are discussed in Section 5.

3.3 Semi-honest protocols

In the semi-honest setting, we use a simple framework to unify the state-of-art concretely efficient MPC protocols, including 1) the GMW protocol [4] with optimizations [102, 113–115] based on additive secret sharing; 2) the BGW protocol [2] with optimizations [116–119] based on Shamir secret sharing; and 3) the secure three-party computation (3PC) protocol [87] based on replicated secret sharing. Here, for MPC based on replicated secret sharing, we focus on the three-party case for the sake of simplicity.

Protocol $\Pi_{\mathsf{MPC}}^{\mathsf{semi}}$

Inputs: Every party P_i ($i \in [1, n]$) holds an input $\mathbf{x}_i \in \mathbb{F}^m$, and also knows a description of a circuit C over a field $\mathbb{F}$. All n parties are desired to compute an output vector $(\mathbf{y}_1, \ldots, \mathbf{y}_n) = C(\mathbf{x}_1, \ldots, \mathbf{x}_n)$.

Protocol execution: $P_1, \ldots, P_n$ interact with each other as follows.

1. *Processing the inputs:* For each input wire w associated with the P_i's input x_w, P_i acts as the dealer to run $\mathsf{Share}(x_w)$ with all other parties, and thus the parties obtain $[x_w]$.
2. *Circuit evaluation:* All parties evaluate the circuit in topological order. For each gate $(\alpha, \beta, \gamma, T)$ with input sharings $[x]$, $[y]$, the parties execute as follows:
   - If $T = \mathsf{ADD}$, the parties compute $[z] := [x] + [y]$ *locally*.
   - If $T = \mathsf{MULT}$, the parties execute a multiplication sub-protocol $\Pi_{\mathsf{Mult}}^{\mathsf{semi}}$ on inputs $[x]$, $[y]$ to compute $[z]$ with $z = x \cdot y \in \mathbb{F}$. The classical multiplication protocols for LSSSs of three kinds are described in Figure 2.
3. *Output reconstruction:* For each output wire w associated with the P_i's output y_w, the parties run $\mathsf{Reconstruct}([y_w], i)$.

Figure 1. Framework for semi-honest MPC protocols based on the secret-sharing approach in both dishonest-majority and honest-majority settings

While the original GMW protocol only considers Boolean circuits, we easily extend it to arithmetic circuits over any finite field $\mathbb{F}$ [120]. Similarly, while the state-of-the-art 3PC protocol [87] with semi-honest security in the honest-majority setting focuses on the case of Boolean circuits, it is easy to extend this protocol to work over any finite field $\mathbb{F}$ [101]. Toward more parties (*e.g.*, the number of parties is $n = 4$ or $n = 5$), MPC protocols based on replicated secret sharing can be constructed efficiently (see, *e.g.*, [124–126]). In the presence of semi-honest adversaries, the GMW-like protocols and the MPC protocols based on replicated secret sharing can be straightforwardly extended to work over a ring such as $\mathbb{Z}_{2^k}$ for $k = 32$ or $k = 64$. Furthermore, the BGW-like protocols based on Shamir secret sharing can also work over a general ring (see, *e.g.*, [127]). While integer computations modulo $\mathbb{Z}_{2^k}$ are more natural for modern computers, and may be useful for simplifying implementations and applications such as machine learning (ML), we focus on the case of finite fields for the sake of simplicity.

Below, we present the framework for secret-sharing-based MPC protocols in the semi-honest setting, which is shown in Figure 1. Specifically, the inputs are shared secretly among all parties, and then the circuit is evaluated layer-by-layer where all gates in a layer can be computed in parallel, and thus the communication round is linear to the depth of the circuit. Finally, the output of every party is reconstructed. While addition gates are free without any communication, the main cost of MPC protocols is to compute multiplication gates by executing a semi-honest multiplication protocol $\Pi_{\mathsf{Mult}}^{\mathsf{semi}}$. For LSSSs of different kinds, $\Pi_{\mathsf{Mult}}^{\mathsf{semi}}$ is constructed in a different way. We sketch three classical constructions of $\Pi_{\mathsf{Mult}}^{\mathsf{semi}}$ corresponding to three kinds of secret sharings in Figure 2, where the protocols are divided into two phases: the preprocessing phase where the circuit and input are unknown and the online phase where the circuit and input are known to all parties.

Toward additive secret sharing, the state-of-the-art protocol adopts Beaver multiplication triples [121] to perform the multiplication of two secret values. In particular, a random Beaver triple $([a], [b], [c])$ with $c = a \cdot b \in \mathbb{F}$ is generated in the preprocessing phase, and then is consumed to compute one multiplication using the Beaver technique in the online phase. If $\mathbb{F} = \mathbb{F}_2$, then Beaver triples can be computed by OT extension protocols; otherwise, they are able to be computed using OLE protocols. Recently, Mouchet *et al.* [128] also used the multi-party homomorphic encryption (MHE) scheme to generate Beaver triples with semi-honest security in the dishonest-majority setting. When the Beaver technique requires two elements per multiplication gate per party in the online phase, the communication can be further reduced to one element per multiplication gate per party using the technique underlying the Turbospeedz protocol [129]. In particular, the function-dependent preprocessing phase is introduced where the circuit is known but the input is still unknown, and circuit-dependent Beaver triples are generated

Protocol $\Pi^{\text{semi}}_{\text{Mult}}$

Inputs: $P_1, \ldots, P_n$ input shares of two secret values $[x]$ and $[y]$, and will compute a random share $[z]$ such that $z = x \cdot y \in \mathbb{F}$.

Additive secret sharing: The multiplication can be efficiently realized using the Beaver technique [121].

1. *Preprocessing phase:* All parties generate a random Beaver triple $[a], [b], [c]$ with $c = a \cdot b \in \mathbb{F}$ by executing a semi-honest either OT or OLE protocol depending on whether $\mathbb{F} = \mathbb{F}_2$ or not.
2. *Online phase:* The parties execute as follows:
   (a) Run $\epsilon := \mathsf{Open}([x] - [a])$ and $\sigma := \mathsf{Open}([y] - [b])$.
   (b) Compute $[z] := [c] + \epsilon \cdot [b] + \sigma \cdot [a] + \epsilon \cdot \sigma$.

Shamir secret sharing: This can be done by executing the DN protocol [103].

1. *Preprocessing phase:* The parties generate a pair of random double sharings $([r]_t, [r]_{2t})$ by executing the protocol [122], where $[x]_d$ denotes a degree-d Shamir secret sharing.
2. *Online phase:* All parties execute the following steps:
   (a) Compute locally $[\epsilon]_{2t} := [x]_t \cdot [y]_t + [r]_{2t} = [x \cdot y + r]_{2t}$, where the shares of $[x]_t \cdot [y]_t$ can be obtained by letting every party multiply its share of $[x]_t$ by its share of $[y]_t$ directly.
   (b) Let $P_{\mathtt{king}}$ be one of $P_1, \ldots, P_n$. Execute $\mathsf{Reconstruct}([\epsilon]_{2t}, P_{\mathtt{king}})$ to enable $P_{\mathtt{king}}$ to obtain $\epsilon = x \cdot y + r \in \mathbb{F}$.
   (c) $P_{\mathtt{king}}$ sends ϵ to all parties, who compute $[z]_t := \epsilon - [r]_t$ locally.

Replicated secret sharing: The multiplication is able to be performed by running the protocol [87, 123].

1. *Preprocessing phase:* All parties P_1, P_2, P_3 generate a random 0-sharing $[0]$ such that P_i obtains r^i for $i \in [1, 3]$ and $r^1 + r^2 + r^3 = 0$ using the pseudorandom function (PRF) [87, 123].
2. *Online phase:* Let (x^{i-1}, x^{i+1}) and (y^{i-1}, y^{i+1}) be the P_i's share of x and y, respectively. The parties run the following steps:
   (a) For $i \in [1, 3]$, P_i computes $z^{i-1} := x^{i-1} \cdot y^{i-1} + x^{i-1} \cdot y^{i+1} + x^{i+1} \cdot y^{i-1} + r^i$ and sends z^{i-1} to P_{i+1}. Note that these messages can be computed and sent in parallel.
   (b) After z^{i-1} has been computed and z^{i+1} was received, every party P_i defines (z^{i-1}, z^{i+1}) as its share of value $z = x \cdot y$.

Figure 2. Semi-honest multiplication protocols based on secret sharings of different types

in this phase. Then, in the online phase, public values (*i.e.*, the values obtained by masking actual values with random values) are computed by the parties, and the multiplication is performed with public values and circuit-dependent Beaver triples. We describe the GMW protocol with this optimization as follows:

(1) Consider $(\Lambda_w, [\lambda_w])$ as the additive secret sharing on a wire w, where $\Lambda_w = z_w + \lambda_w \in \mathbb{F}$ is a public value, z_w is the actual value on w and λ_w is a random element. Here, $[\lambda_w]$ can be generated by having every party share a random value to other parties and then sum all its shares as the resulting share of λ_w in the preprocessing phase.
(2) For each wire w associated with P_i's input x_w, P_i sends $\Lambda_w = x_w + \lambda_w$ to all parties who define $(\Lambda_w, [\lambda_w])$ as the sharing on wire w.
(3) For each addition gate $(\alpha, \beta, \gamma, \mathsf{ADD})$ with input sharings $(\Lambda_\alpha, [\lambda_\alpha])$ and $(\Lambda_\beta, [\lambda_\beta])$, the parties locally compute $\Lambda_\gamma = \Lambda_\alpha + \Lambda_\beta \in \mathbb{F}$ and $[\lambda_\gamma] = [\lambda_\alpha] + [\lambda_\beta]$, where $[\lambda_\gamma]$ can be computed in the function-dependent preprocessing phase.
(4) For each multiplication gate $(\alpha, \beta, \gamma, \mathsf{MULT})$ with input sharings $(\Lambda_\alpha, [\lambda_\alpha])$ and $(\Lambda_\beta, [\lambda_\beta])$, all parties generate a random sharing $[\lambda_{\alpha\beta}]$ with $\lambda_{\alpha\beta} = \lambda_\alpha \cdot \lambda_\beta \in \mathbb{F}$ by running a semi-honest

OT/OLE protocol with inputs $[\lambda_\alpha], [\lambda_\beta]$ in the function-dependent preprocessing phase. Then, the parties compute

$$[\Lambda_\gamma] := \Lambda_\alpha \cdot \Lambda_\beta - \Lambda_\alpha \cdot [\lambda_\beta] - \Lambda_\beta \cdot [\lambda_\alpha] + [\lambda_{\alpha\beta}] + [\lambda_\gamma],$$

where $[\lambda_\gamma]$ is a random sharing generated by the parties in the preprocessing phase. The parties execute $\mathsf{Open}([\Lambda_\gamma])$ to obtain Λ_γ, and define $(\Lambda_\gamma, [\lambda_\gamma])$ as the sharing on the output wire γ.

The GMW protocol based on the multiplication protocol $\Pi^{\mathsf{semi}}_{\mathsf{Mult}}$ shown in Figure 2 supports the corruption threshold $t = n - 1$ with n the number of total parties. If we allow $n/2 \leq t < n - 1$ (that is still in the dishonest-majority setting), we can use the TinyKeys approach [114] to improve the efficiency of protocol $\Pi^{\mathsf{semi}}_{\mathsf{Mult}}$ over a binary field. Specifically, for the number of honest parties $h > 1$, *e.g.*, $(h = 6, n = 20)$ or $(h = 120, n = 400)$, Hazay *et al.* [114] used the IKNP OT extension protocol [113, 130] with short keys to generate Beaver multiplication triples. The basic idea is that the combination of the short keys of h honest parties will have a large entropy, although the short key of an honest party has only a low entropy. The TinyKeys approach can also be extended to the malicious setting using IT-MACs with short keys [131].

Due to the recent development of PCG-style OT extension protocols [132–134], these protocols outperform the IKNP-style OT extension protocols [113, 130, 135, 136], and have the sublinear communication compared to the linear communication of the IKNP-style protocols (see Section 5 for more details). In this case, the efficiency improvement using the TinyKeys approach seems to be significantly smaller, if the recent PCG-style (instead of IKNP-style) OT extension protocols are adopted to construct the protocol $\Pi^{\mathsf{semi}}_{\mathsf{Mult}}$.

For Shamir secret sharing, if the number of parties is small (particularly $n \leq 5$), the state-of-the-art multiplication protocol is the GRR protocol [117] using the degree-reduction approach. The GRR protocol works by letting every party locally multiply its shares of the inputs $[x], [y]$, share the result to all other parties (allowing the degree of the polynomial to be reduced from $2t$ to t), and then locally compute a linear combination of shares as its share of the output $[z]$. If the number of parties is larger (*e.g.*, $n > 5$), we mainly adopt the Damgård–Nielsen (DN) protocol [116] to realize the multiplication of two secret values. The original DN protocol [116] described in Figure 2 is information-theoretically secure, and needs six elements of communication per multiplication gate per party. In the information-theoretic setting, the communication cost of the DN protocol was first improved by Goyal *et al.* [119, 137] from 6 elements to 5.5 elements using a simple technique. Recently, the communication of the DN protocol was further improved to 4 elements per multiplication gate per party by Goyal *et al.* [118]. Inspired by the technique [119, 137], they first modify the original DN protocol [116] shown in Figure 2 to make $P_{\texttt{king}}$ send a random degree-t Shamir sharing $[\epsilon]_t$ (instead of ϵ) to all other parties and then all parties compute $[z]_t := [\epsilon]_t - [r]_t$ locally. Their crucial observation is that when $P_{\texttt{king}}$ is an honest party, the corrupted parties only receive random elements from $P_{\texttt{king}}$ in this case. This holds even if the corrupted parties know the whole double sharings $([r]_t, [r]_{2t})$, since they only receive t shares that are uniformly random and independent of secret $z = x \cdot y$. Therefore, we can split the tasks of computing multiplication gates as $P_{\texttt{king}}$ into all parties rather than a fixed party in the original DN protocol. In particular, when we need to compute n multiplication gates, we let every party behave as $P_{\texttt{king}}$ for one multiplication gate. If $P_{\texttt{king}}$ is a corrupted party, we still need a pair of random double sharings. If $P_{\texttt{king}}$ is an honest party, the double sharings do not need to be random. This means that we only need to generate t pairs of random double sharings for n multiplication gates in the preprocessing phase. In the computation setting, we can use the pseudorandom secret sharing approach [99] to further reduce the communication cost, which has been used in previous honest-majority MPC protocols such as [101, 138–140]. For example, the state-of-the-art DN-style protocol [118] can be optimized to two elements per multiplication gate per party using a pseudo-random generator (PRG). In addition, Goyal *et al.* [118] also proposed a combination of the improved DN multiplication and Beaver triple multiplication to reduce the round complexity by a factor of 2, at the cost of that the communication cost is additionally increased by 0.5 elements per multiplication gate per party. Recently, Abspoel *et al.* [141] used regenerating codes to construct a single-round multiplication protocol at the cost of increasing the communication complexity by a factor $O(n/\log n)$, where the regenerating property [142] of Shamir secret sharing requires that the number of parties n is large and the DN multiplication protocol needs about two rounds.

As for replicated secret sharing in the 3PC setting, the state-of-the-art protocol [87, 101] is simple and needs to send only one element per multiplication gate per party. In particular, shares of $z = x \cdot y$ can

be computed by every party locally. Then, every party needs to use a random 0-sharing to randomize its share of z, and then sends the result to another party. The random 0-sharing $[r]$ is easy to be computed by having every party P_i send a uniform key k_i to P_{i+1} for $i \in [1,3]$ and compute $r^i := F(\mathsf{k}_{i-1}, id) - F(\mathsf{k}_i, id)$ using the two keys that it holds where id is an identifier. While Shamir secret sharing is suitable for a large number of parties, replicated secret sharing is mainly used for a small number of parties (*e.g.*, the number of parties $n \in \{3,4,5,7,9\}$). In addition, Keller *et al.* [143] showed that the 3PC protocol [87] can be generalized to a general Q_2 access structure, which was further improved in [144] to eliminate the restriction of replicated secret sharing (*i.e.*, requiring an exponentially-large number of shares for a large number of parties).

3.4 Maliciously secure protocols

The MPC protocols based on secret sharings described in the previous subsection guarantee security in the presence of semi-honest adversaries. To achieve malicious security, some checking procedures need to be added. The underlying techniques to assure security against malicious adversaries are different between dishonest-majority MPC and honest-majority MPC. For example, MPC in the dishonest-majority setting needs IT-MACs to authenticate values shared among all parties, but this is unnecessary for MPC with honest majority. Thus, we present the development for maliciously secure MPC in two different settings.

Dishonest majority. Goldreich, Micali and Wigderson (GMW) [4] proposed a general compiler to convert a semi-honest MPC protocol into a maliciously secure MPC protocol for the same computational task. However, this compiler is non-black-box using the generic zero-knowledge proofs to prove the correctness of computation in each step, and thus is not concretely efficient. Later, Ishai, Prabhakaran and Sahai (IPS) [145] proposed a black-box compiler, where an inner MPC protocol with semi-honest security computes a circuit in the OT-hybrid model, and an outer MPC protocol with malicious security in the honest-majority setting is used to guarantee the security of the whole MPC protocol in the presence of malicious adversaries. The IPS compiler was improved in [123] for multi-party setting, and was further optimized in [54, 146] for two-party setting. However, the concrete efficiency for the maliciously secure MPC protocols based on the IPS compiler is still not sufficiently high. Recently, based on the IPS framework, Hazay *et al.* [147] proposed a new compiler using two-level sharings where the outer level is Shamir secret sharing or algebraic geometric (AG) secret sharing [148], and the inner level is additive secret sharing. Their compiler allows an arbitrary-sized field with constant communication overhead over the semi-honest GMW protocol [4], but the concrete efficiency is still low.

In the dishonest-majority setting, concretely efficient MPC protocols based on IT-MACs have the smallest overhead to achieve malicious security. Using IT-MACs, we can transform the semi-honest GMW protocol shown in Figure 1 to a maliciously secure protocol in the following way:

- Replacing all additive secret sharings with authenticated secret sharings defined in Section 3.2.
- For each wire w associated with P_i's input x_w, the parties generate a random authenticated share $[\![r_w]\!]$ in the preprocessing phase, and then P_i broadcasts $\Lambda_w := x_w - r_w$ to all parties who compute $[\![x_w]\!] := [\![r_w]\!] + \Lambda_w$.
- Replacing the semi-honest multiplication protocol $\Pi_{\mathsf{Mult}}^{\mathsf{semi}}$ with a maliciously secure protocol $\Pi_{\mathsf{Mult}}^{\mathsf{mali}}$.

We can also use the Beaver technique [121] to construct protocol $\Pi_{\mathsf{Mult}}^{\mathsf{mali}}$ in the following steps:

- *Preprocessing phase:* All parties generate a random authenticated triple $([\![a]\!], [\![b]\!], [\![c]\!])$ with $c = a \cdot b \in \mathbb{F}$. The generation of authenticated triples can be divided into two steps: 1) computing authenticated shares by executing a correlated OT (COT) or vector OLE (VOLE) protocol with malicious security, where the notion of COT and VOLE can be found in Section 5; 2) generating faulty authenticated triples with authenticated shares and then checking the correctness of these faulty authenticated triples. In the first step, all parties execute a maliciously secure COT/VOLE protocol in a pairwise way, and then run a consistency-check procedure to check the consistency of shares and global keys among multiple executions. The state-of-the-art consistency check adopts the random linear combination approach (see, *e.g.*, [43, 104, 106, 108, 110]), and requires a very small communication overhead. For the second step, multiple approaches can be used for different applications (see below).

- *Online phase:* This phase is similar to the semi-honest protocol shown in Figure 2, except that authenticated shares are involved and the corresponding open procedure described in Section 3.2 is used. Similarly, the communication in this phase can be further improved from 2 elements per multiplication gate per party to only one element, using the technique in Turbospeedz [129].

When the Turbospeedz technique [129] and the batch-check technique shown in Section 3.2 are used, the online phase of the maliciously secure GMW-like protocols [103, 104] has the optimal communication cost without sacrificing the security. Most of MPC studies focus on improving the efficiency of the preprocessing phase. Particularly, generating authenticated triples is the efficiency bottleneck. For the generation of authenticated triples, we consider two cases: 1) large fields (*i.e.*, $|\mathbb{F}| \geq 2^\rho$) and 2) binary fields (*i.e.*, $\mathbb{F} = \mathbb{F}_2$). Note that the techniques for binary fields are able to be used in other cases of small fields (*e.g.*, $\mathbb{F}_{2^8}$).

For the case of large fields, the SPDZ framework [103, 149] is the state-of-the-art protocol in the dishonest-majority malicious setting. The original SPDZ protocol [103, 149] uses the depth-1 homomorphic encryption (HE) scheme (*i.e.*, the underlying HE scheme could support one multiplication) to generate authenticated triples in the preprocessing phase, and can evaluate the circuit fast in the online phase. Later, Keller *et al.* [43] proposed a SPDZ-style protocol called as MASCOT, which uses the OT extension protocol [136] and Gilboa multiplication idea [150] to generate authenticated triples more efficiently. Subsequently, based on additively homomorphic encryption [151] and lattice-based zero-knowledge proofs, Keller *et al.* [152] presented an optimized SPDZ-style protocol referred to as Overdrive, which significantly improves the communication to generate authenticated triples. Overdrive includes two versions: LowGear for a small number of parties and HighGear for a large number of parties. The performance of HighGear is further improved by Baum *et al.* [153] *via* optimizing the underlying zero-knowledge protocol. In these SPDZ protocols, the underlying technique for checking the correctness of faulty authenticated triples is the so-called "sacrifice" technique. The improved sacrifice technique [43] works as follows:

- Let $(\llbracket a \rrbracket, \llbracket b \rrbracket, \llbracket c \rrbracket)$ and $(\llbracket \hat{a} \rrbracket, \llbracket b \rrbracket, \llbracket \hat{c} \rrbracket)$ be two faulty authenticated triples held by parties $P_1, \ldots, P_n$. The parties execute the following procedure to check the correctness of the first triple $(\llbracket a \rrbracket, \llbracket b \rrbracket, \llbracket c \rrbracket)$ by sacrificing the second triple $(\llbracket \hat{a} \rrbracket, \llbracket b \rrbracket, \llbracket \hat{c} \rrbracket)$.
  (a) Run a coint-tossing protocol to sample a uniformly random element $r \in \mathbb{F}$.
  (b) Compute $\llbracket \epsilon \rrbracket := r \cdot \llbracket a \rrbracket - \llbracket \hat{a} \rrbracket$ locally.
  (c) Execute $\mathsf{Open}(\llbracket \epsilon \rrbracket)$ to obtain ϵ.
  (d) Compute $\llbracket \sigma \rrbracket := r \cdot \llbracket c \rrbracket - \llbracket \hat{c} \rrbracket - \epsilon \cdot \llbracket b \rrbracket$ locally.
  (e) Run $\mathsf{CheckZero}(\llbracket \sigma \rrbracket)$ to verify that $\sigma = 0$, where $\mathsf{CheckZero}$ is the same as Open defined in Section 3.2, except that the values to be opened are 0 and thus are unnecessary to be sent.
- When ℓ faulty authenticated triples need to be checked for some integer ℓ, the randomness r can be reused for all ℓ checking procedures, and the procedures Open and $\mathsf{CheckZero}$ can be done in a batch (see Section 3.2).
- If $c = a \cdot b + e$ for some adversarially chosen error e and $e \neq 0$, then we have the following:

$$\begin{aligned} \sigma &= r \cdot c - \hat{c} - \epsilon \cdot b \\ &= r \cdot (a \cdot b + e) - \hat{c} - (r \cdot a - \hat{a}) \cdot b \\ &= r \cdot e - \hat{e}, \end{aligned}$$

  where $\hat{e} = \hat{c} - \hat{a} \cdot b$ is another error introduced by the adversary. If $e \neq 0$, the probability that $\sigma = r \cdot e - \hat{e}$ is equal to 0 is $1/|\mathbb{F}|$, as r is sampled uniformly at random after $e, \hat{e}$ have been determined. Therefore, the checking procedure described as above requires $\mathbb{F}$ to be a large field. We can also repeat the checking procedure t times to support a small field $\mathbb{F}$, where $|\mathbb{F}|^t \geq 2^\rho$ and $t = \rho$ if $\mathbb{F} = \mathbb{F}_2$. However, this will require a large computation and communication overhead.

Recently, Chen *et al.* [154] integrated the depth-2 HE scheme [155, 156] into the SPDZ framework to improve the efficiency of SPDZ protocols for computing matrix multiplication and two-dimensional convolution. For other general functions, it is not clear whether their approach can achieve a better efficiency, due to the larger parameters for HE.

While the semi-honest GMW protocol over a field can be straightforwardly extended to ring $\mathbb{Z}_{2^k}$, this is not easy for SPDZ-style protocols with malicious security. Cramer *et al.* [157] proposed the first concretely

efficient MPC protocol over a ring $\mathbb{Z}_{2^k}$ in the SPDZ framework (named as SPD$\mathbb{Z}_{2^k}$), using IT-MACs over two different rings where the secret values are in $\mathbb{Z}_{2^k}$ and the MAC tags are in $\mathbb{Z}_{2^{k+s}}$. Their protocol SPD$\mathbb{Z}_{2^k}$ uses the MASCOT idea to generate authenticated triples, but needs more communication than MASCOT [43]. Later, Damgård *et al.* [158] implemented SPD$\mathbb{Z}_{2^k}$, designed new protocols for equality test, comparison, and truncation over ring $\mathbb{Z}_{2^k}$, and demonstrated that these operations in the ML domain using SPD$\mathbb{Z}_{2^k}$ are more efficient than the field-based SPDZ-style protocols [43, 152]. Subsequently, two improved MPC protocols based on the SPD$\mathbb{Z}_{2^k}$ idea have also been proposed [159, 160].

Recently, Boyle *et al.* [161] proposed several new protocols to generate Beaver multiplication triples and authenticated triples in the PCG framework based on a new variant of the ring-LPN assumption. They use distributed point function (DPF) and the sparse feature of the noise in ring-LPN to generate Beaver triples in the semi-honest two-party setting with a small communication. Using the programmability of PCG [162], the semi-honest protocol for producing Beaver triples in the two-party setting can be easy to be extended to the multi-party setting. Based on the construction of SPDZ-style authenticated shares, Boyle *et al.* [161] also extended the semi-honest protocol to generate authenticated triples in the two-party malicious setting. The maliciously secure protocol has a communication that is two orders of magnitude smaller than Overdrive. While the communication efficiency is attractive, it is worth further reducing the computational cost to make the PCG approach more practical. Boyle *et al.* [161] also gave a candidate construction for generating authenticated triples in the multi-party setting using the three-party DPF [163], but its concrete efficiency is very low.

For the case of binary field (*i.e.*, $\mathbb{F} = \mathbb{F}_2$), we mainly consider two types of MPC protocols in the malicious setting: TinyOT-style [104, 106–110, 164, 165] and MiniMAC-style [164, 166–169]. In particular, the sub-protocols for generating authenticated triples underlying in the TinyOT-style protocols can be used to design constant-round MPC protocols with malicious security [106–110]. These TinyOT-style protocols adopt the BDOZ-style IT-MACs [105] to authenticate bits, and use the bucketing approach to eliminate the possible leakage of shares due to the selective-failure attack, where the adversary can guess a bit share of an honest party with probability $1/2$ but will be caught with the same probability. MiniMAC [168] aims to solve the problem that SPDZ [103] has a large communication overhead for binary field $\mathbb{F}_2$ (particularly the communication is blown up by a factor of ρ). Specifically, MiniMAC adopts a batch authentication idea: if k instances of the same Boolean circuit need to be computed at once, one can bundle these computations together and view them as the computation of a single Boolean circuit over a large ring $\mathbb{F}^k$, where the addition and multiplication over $\mathbb{F}^k$ are component-wise. In MiniMAC, an IT-MAC on message $\mathbf{x} \in \mathbb{F}^k$ is defined as $C(\mathbf{x}) * \boldsymbol{\Delta}$ where $\boldsymbol{\Delta} \in \mathbb{F}^k$, C is a linear error-correcting code with a large minimum distance and $*$ is the component-wise product. MiniMAC-style MPC protocols [164, 166–169] also work for layered Boolean circuits where the gates of a Boolean circuit are partitioned into ordered layers and a layer only consists of gates of the same type. The recent MiniMAC-style protocol [166] adopts an algebraic tool called reverse multiplication friendly embedding (RMFE) [170] that is originally proposed for honest-majority MPC, and obtains a lower communication cost. Besides, for small fields, TinyTable-style protocols [112, 171, 172] are proposed and very suitable for secure AES or 3DES evaluation using the one-time truth table approach.

Honest majority. In the malicious setting, we only need to check the correctness of multiplication gates, as addition gates are computed locally and always correct. In 2017, Lindell and Nof [101] observed that the semi-honest DN protocol [116] has guaranteed the privacy of secret values in the presence of malicious adversaries, and allows the adversary to introduce an additive error in the output, *i.e.*, for two sharings $[x], [y]$, the DN protocol will output a sharing $[z]$ with $z = x \cdot y + d$ where d is an additive error. This observation also holds for the GRR protocol [117] and the multiplication protocol based on replicated secret sharing [87]. They adopt the Beaver triples and the random-linear-combination approach to check the correctness of multiplication gates, which introduces a relatively large overhead compared to the semi-honest protocol. Subsequently, Chida *et al.* [138] proposed a different approach to verify the correctness of multiplication gates, where the semi-honest multiplication protocol is executed twice and then the parties check the correctness of a multiplication gates using another related multiplication triple. Their MPC protocol still introduces twice the communication overhead compared to the semi-honest DN protocol. Concurrently, Nordholt and Veeningen [140] also achieved the twice communication overhead. The studies [101, 138] mainly consider the case of large fields, and also present the correctness check

Table 1. Comparison of honest-majority MPC protocols based on Shamir secret sharing

Semi-honest MPC	Comm. per mult. gate per party	Maliciously secure MPC	Comm. overhead
[116]	6 elements (2.5 elements)	[101]	7×
[119, 137]	5.5 elements (2.5 elements)	[138, 140]	2×
[118]	4 elements (2 elements)	[125, 137]	$1 + o(1)$ ×

Note: We compare the communication cost for evaluating a single arithmetic circuit. We use "Comm." to denote communication and "mult." to denote multiplication. If the values are in parenthesis "()", then they represent the communication costs in the computational setting using the PRSS approach, otherwise they denote the communication costs in the information-theoretic setting.

for small fields by repeating the verification procedure which will introduce an large overhead. In the three-party setting, Furukawa *et al.* [173] and Araki *et al.* [174] converted the semi-honest protocol for Boolean circuits [87] to maliciously secure protocols using the "Cut-and-Choose" approach, which will introduce a overhead of $O(\rho/\log N)$ where N is the number of multiplication gates. This overhead is smaller than the natural repeat approach, but is not optimal.

The MPC protocols described as above allow the corruption threshold $t < n/2$ where n is the number of parties. If $t < n/3$ is allowed, the MPC protocol with malicious security can be constructed at essentially the same cost as the best-known semi-honest protocol [139], *i.e.*, the overhead to achieve malicious security is 1 and optimal. Their approach is as follows: 1) for two Shamir sharings $[x]_t$ and $[y]_t$, the parties can locally compute the Shamir sharing of $[z]_{2t}$ with $z = x \cdot y$; 2) when $t < n/3$, the opening of $[z]_{2t}$ can be guaranteed to be correct; 3) $[z]_{2t}$ can be used to check the correctness of $[z]_t$ that is obtained by running the semi-honest DN protocol. If $t < n/2$, there are two recent techniques to achieve malicious security with an overhead of 1 over the best-known semi-honest protocol.

Specifically, one can use the distributed zero-knowledge proof with sublinear communication [124] to verify the correctness of multiplication gates. Firstly, Boneh *et al.* [124] used such zero-knowledge proofs and a variant of the DN protocol with replicated secret sharing to construct a maliciously secure MPC protocol for constant number of parties. At the first time, their approach obtains 1 bit per AND gate per party in terms of the amortized communication cost for the 3PC protocol about Boolean circuits. Their verification protocol requires $O(n\sqrt{N}+n)$ field elements per party of communication and constant rounds using the Fiat–Shamir heuristic, where n is the number of parties. In the three-party setting, the concrete efficiency of the 3PC protocol by Boneh *et al.* [124] is significantly improved in [175], which achieves the best efficiency for now. Recently, Boyle *et al.* [125] used the distributed zero-knowledge proof in a new way, and constructed an MPC protocol with an optimal overhead over the best-known semi-honest protocol for an arbitrary number of parties. Their approach uses a new insight where for any secret sharing of a value x, we can simultaneously view shares of x as a sharing of each secret share x^i itself. Their verification protocol [125] for checking multiplication triples needs communication of $O(n \log N + n)$ field elements per party and constant rounds.

Building on the technique [124] of distributed zero-knowledge proofs, Goyal *et al.* [119, 137] proposed another verification technique for multiplication gates to achieve malicious security with an overhead of 1, and requires $O(\log N)$ rounds and communication of $(n \log N + n)$ field elements per party for the verification protocol.

They used a key observation that the semi-honest DN protocol can compute the inner-product of two vectors with the same communication cost [138], and adopted a recursive idea to perform the verification of multiplication gates. Concretely, their verification technique works as follows:

(1) Given $N = |\mathcal{C}|$ multiplication triples $\{([x_i], [y_i], [z_i])\}_{i\in[1,N]}$ to be verified, where these tuples are computed using the semi-honest DN protocol, parties $P_1, \ldots, P_n$ use a random-linear-combination approach with a uniformly random r to compute the following vectors:

$$
\begin{aligned}
[\mathbf{x}] &:= ([x_1], r \cdot [x_2], \ldots, r^{N-1} \cdot [x_N]) \\
[\mathbf{y}] &:= ([y_1], [y_2], \ldots, [y_N]) \\
[z] &:= \sum_{i\in[1,N]} r^{i-1} \cdot [z_i].
\end{aligned}
$$

The verification of multiplication triples is reduced to verify whether $z = \mathbf{x} \cdot \mathbf{y}$ for $([\mathbf{x}], [\mathbf{y}], [z])$.

(2) Let k be a compression parameter. The parties transform the original tuple of dimension N into a new tuple of dimension N/k. Rewrite $[\mathbf{x}]$ and $[\mathbf{y}]$ as follows:

$$[\mathbf{x}] = ([\mathbf{a}_1], [\mathbf{a}_2], \ldots, [\mathbf{a}_k])$$
$$[\mathbf{y}] = ([\mathbf{b}_1], [\mathbf{b}_2], \ldots, [\mathbf{b}_k]),$$

where $\{\mathbf{a}_i, \mathbf{b}_i\}_{i\in[1,k]}$ are vectors of dimension N/k. For $i \in [1, k-1]$, the parties execute the semi-honest DN protocol with inner-product extension to compute $[c_i]$ with $c_i = \langle \mathbf{a}_i, \mathbf{b}_i \rangle$. Then, they set $[c_k] = [z] - \sum_{i\in[1,k-1]}[c_i]$. Now, the parties need to verify the correctness of $([\mathbf{a}_i], [\mathbf{b}_i], [c_i])$ for $i \in [1, k]$. This can be done in a batch using the batch-wise multiplication verification technique [140, 176] to compress the verification of k inner-product tuples into one check of a single inner-product tuple of dimension m/k.

(3) The parties repeat the second step $\log_k N$ times so that only a single multiplication triple needs to be checked. This final check can be performed using the randomization and opening approach.

The verification technique by Goyal *et al.* [119, 137] is originally described for Shamir secret sharing, and can also work for replicated secret sharing as the state-of-the-art semi-honest multiplication protocol [87, 101] has the same communication cost for computing inner-product of two vectors.

Both of the state-of-the-art verification techniques [125, 137] are based on the technique underlying the distributed zero-knowledge proofs [124]. Both techniques can obtain the same communication complexity, but has a different round complexity where the technique [119] requires $O(\log N)$ of rounds and the technique [125] has constant rounds. In terms of concrete communication efficiency, the verification protocol of multiplication gates by Goyal *et al.* [137] is slightly better than the protocol by Boyle *et al.* [125].

In Table 1, we compare the communication cost of several known honest-majority MPC protocols based on Shamir secret sharing for evaluating a single arithmetic circuit, where the left part compares the communication cost of semi-honest MPC protocols and the right part compares the communication overhead of maliciously secure MPC protocols over semi-honest protocols. For a large number of parties, the honest-majority MPC protocols as described above adopt Shamir secret sharing as the underlying LSSS, and thus require $O(n \log n)$ bits per multiplication gate of communication complexity for evaluating a single Boolean circuit, as Shamir secret sharing requires that the size of field $\mathbb{F}$ is greater than the number n of parties. Recently, based on RMFE [170], Polychroniadou and Song [177] combined Shamir secret sharing with additive secret sharing to reduce the communication complexity to $O(n)$ bits per multiplication gate.

Two recent studies [178, 179] designed large-scale MPC protocols, which scale practically to hundreds of thousands of parties. Such MPC protocols are interesting for applications that a large number of parties participate in the protocol execution. For example, in privacy-preserving federated learning, thousands of low-resource devices are desired to train a ML model on their collective data. Additionally, when the number of parties is larger, the honest-majority assumption will become more believable. While the honest-majority MPC protocols [101, 116, 118, 119, 125, 137–139] have a total communication complexity $O(n|\mathcal{C}|)$, both concretely efficient MPC protocols [178, 179] adopt packed secret sharing [180] to obtain the total communication complexity $O(|\mathcal{C}|)$, where n is the number of parties and $|\mathcal{C}|$ is the size of the circuit to be computed. While the work by Gordon *et al.* [179] has the total computation complexity $O(\log n \cdot |\mathcal{C}|)$ for any polynomial-sized circuit, the work by Beck *et al.* [178] obtains the total computation complexity $O(|\mathcal{C}|)$ for highly repetitive circuits (*e.g.*, ML training algorithms). Packed secret sharing is an important tool that has been used to obtain the *total* communication complexity $\tilde{O}(|\mathcal{C}|)$ for SIMD circuits[1] [181–183], and is a generalization of Shamir secret sharing that defines as follows:

- Let n be the number of parties and k be the number of secrets that are packed in a single sharing. Let $\alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_k$ be $n + k$ distinct non-zero elements over a field $\mathbb{F}$.
- A degree-d packed Shamir sharing of secret vector $\mathbf{x} = (x_1, \ldots, x_k) \in \mathbb{F}^k$ is a vector $(y^1, \ldots, y^n)$, which satisfies that there exists a polynomial $f(\cdot) \in \mathbb{F}[X]$ of degree d, such that for each $i \in [1, k]$, $f(\beta_i) = x_i$ and for all $i \in [1, n]$, $f(\alpha_i) = y^i$. Party P_i obtains the ith share y^i.

[1] SIMD circuits are arithmetic circuits that simultaneously evaluate multiple copies of the same circuit on different inputs.

Protocol $\Pi_{\mathsf{2PC}}^{\mathsf{semi}}$

1. **Preprocessing phase:** A garbler $\mathsf{P_A}$ constructs a garbled circuit $\mathcal{GC}$, and sends it along with the decoding information d to an evaluator $\mathsf{P_B}$. In the procedure of generating $\mathcal{GC}$, $\mathsf{P_A}$ computes $(\mathsf{L}_{w,0}, \mathsf{L}_{w,1})$ for each circuit-input (resp., circuit-output) wire w as the encoding (resp., decoding) information e (resp., d), where $\mathsf{L}_{w,0}$ is called 0-label used to encode bit 0 and the 1-label $\mathsf{L}_{w,1}$ can encode bit 1.
2. **Online phase (input processing):** Let x and y denote the inputs of $\mathsf{P_A}$ and $\mathsf{P_B}$, respectively. Let $[1, \ell]$ be the wire indices on the $\mathsf{P_A}$'s input and $[\ell+1, 2\ell]$ denote the wire indices on the $\mathsf{P_B}$'s input.
   (a) $\mathsf{P_A}$ sends $\mathsf{L}_{1,x_1}, \ldots, \mathsf{L}_{\ell,x_\ell}$ to $\mathsf{P_B}$.
   (b) For each $i \in [\ell+1, 2\ell]$, $\mathsf{P_A}$ and $\mathsf{P_B}$ execute an OT extension protocol in which $\mathsf{P_A}$ inputs $(\mathsf{L}_{i,0}, \mathsf{L}_{i,1})$ and $\mathsf{P_B}$ inputs a bit y_i. Then, $\mathsf{P_B}$ obtains labels L_{i,y_i} for all $i \in [\ell+1, 2\ell]$.
3. **Online phase (output processing):** $\mathsf{P_B}$ uses $(\{\mathsf{L}_{i,x_i}\}_{i\in[1,\ell]}, \{\mathsf{L}_{i,y_i}\}_{i\in[\ell+1,2\ell]})$ to evaluate $\mathcal{GC}$ to obtain L_{w,z_w} for each circuit-output wire w. Then, for each circuit-output wire w, $\mathsf{P_B}$ computes an output bit z_w by comparing L_{w,z_w} with $\mathsf{L}_{w,0}, \mathsf{L}_{w,1}$ involved in d.

Figure 3. Semi-honest Yao's 2PC protocol

- To reconstruct the packed secret vector $\mathbf{x}$, at least $d+1$ shares are needed and it can be done by Lagrange interpolation. For a random degree-d packed Shamir sharing of $\mathbf{x}$, any $d-k+1$ shares are independent of secret $\mathbf{x}$. Thus, for corruption threshold t, we have $d \leq t+k-1$.

Recently, Goyal *et al.* [184] constructed a large-scale MPC protocol, which achieves the total communication complexity $O(|\mathcal{C}|)$ for a single circuit evaluation, using Hall's Marriage Theorem. In the malicious setting, the MPC protocol by Goyal *et al.* [184] can achieve an overhead of 1 over the semi-honest protocol using the verification technique [119], while the overhead for other recent MPC protocols [178, 179] is more than twice. Since no implementation is provided, the concrete efficiency of their MPC protocol is not clear. All the recent large-scale MPC protocols [178, 179, 184] require that the number of corrupted parties $t \leq n(1/2-\epsilon)$ for $0 < \epsilon < 1/2$. Besides, Gordon *et al.* [179] suggested to use the SPDZ technique and a committee of $t+1$ parties to design the online protocol, which is usable to reduce the online communication cost as only $t+1$ parties instead of n parties run the online protocol. Concurrently, Escudero and Dalskov [185] improved the online communication cost of honest-majority MPC protocols using the Turbospeedz technique and the committee idea, and obtained the minimal online communication (*i.e.*, one field element per multiplication gate per party).

4 Constant-round MPC based on garbled circuits

For now, known concretely efficient constant-round MPC protocols are constructed based on garbled circuits that are encrypted versions of circuits and can be computed only once. We first consider semi-honest protocols, and then show how they are compiled to maliciously secure MPC protocols.

4.1 Semi-honest protocols

4.1.1 Secure two-party computation

The first constant-round secure two-party computation (2PC) protocol was proposed by Yao [6], and achieves semi-honest security. The Yao's 2PC protocol adopts garbled circuit (GC) and OT as the building blocks. Specifically, using a garbling scheme, a garbler $\mathsf{P_A}$ is able to generate a garbled circuit $\mathcal{GC}$, an encoding information e and a decoding information d.

Then, an evaluator $\mathsf{P_B}$ can evaluate $\mathcal{GC}$ with e, and then obtains the output bits according to d. The garbling scheme enables $\mathsf{P_B}$ to obtain a function output, but does not reveal any other information on the input to $\mathsf{P_B}$. We refer the reader to [186, 187] for the formal definition of garbling schemes. Roughly, Yao's 2PC protocol with semi-honest security is described in Figure 3.

The 2PC protocol can be further optimized using the precomputing OT idea [188], where a random oblivious transfer (ROT) protocol is run in the preprocessing phase, and transform ROT to standard OT with chosen choice bits in the online phase. Besides, a GC can be sent in a pipelined way (*i.e.*, garbled rows for a batch of gates are computed and communicated, and then these are done for the next batch of gates) [189], which allows the GC implementation to scale to an unlimited number of gates using a nearly constant memory. Subsequent studies focus on optimizing the Yao's 2PC protocol in two aspects: improving the construction of GCs and designing more efficient OT extension protocols. Below, we describe the development of GCs, and postpone that of OT extension to Section 5.

Garbled circuits. The first GC construction was introduced by Yao [6] in 1986, but its formal description and security proof was first presented by Lindell and Pinkas until 2004 [190]. The original Yao GC construction requires 8κ bits per gate. The communication cost can be reduced to 4κ bits per gate using the "point-permute" technique [1, 191], where the actual bit z_w on each wire w is masked by a random bit (a.k.a., permute bit) λ_w, and the resulting public value $\Lambda_w = z_w \oplus \lambda_w \in \{0,1\}$ allows to be known by the evaluator. In this case, the decoding information d can be defined as the wire masks λ_w for all circuit-output wires w. The garbled row reduction (GRR) technique [192] is able to further reduce the communication to 3κ bits per gate, where a garbled row is always defined as zero by specially setting the 0-labels. Later, Pinkas *et al.* [193] used the polynomial-interpolation approach to further reduce the communication to 2κ bits per gate, where the technique is called 4-to-2 GRR compared to the 4-to-3 GRR technique [192]. In 2008, Kolesnikov and Schneider [194] proposed the free-XOR technique that enables XOR gates in the circuit to be garbled with no communication. In particular, the garbler sets $\mathsf{L}_{w,1} = \mathsf{L}_{w,0} \oplus \Delta$ for each wire w where Δ is a fixed offset (a.k.a., global key). For each XOR gate $(\alpha, \beta, \gamma, \oplus)$, the garbler also sets $\mathsf{L}_{\gamma,0} = \mathsf{L}_{\alpha,0} \oplus \mathsf{L}_{\beta,0}$ and $\lambda_\gamma = \lambda_\alpha \oplus \lambda_\beta$. Given the public-value and label pairs $(\Lambda_\alpha, \mathsf{L}_{\alpha,\Lambda_\alpha})$ and $(\Lambda_\beta, \mathsf{L}_{\beta,\Lambda_\beta})$ on the input wires, the evaluator can compute locally the public value

$$\Lambda_\gamma := (z_\alpha \oplus z_\beta) \oplus \lambda_\gamma = (\Lambda_\alpha \oplus \lambda_\alpha) \oplus (\Lambda_\beta \oplus \lambda_\beta) \oplus \lambda_\gamma = \Lambda_\alpha \oplus \Lambda_\beta$$

and the label

$$\mathsf{L}_{\gamma,\Lambda_\gamma} := \mathsf{L}_{\gamma,0} \oplus \Lambda_\gamma \cdot \Delta = (\mathsf{L}_{\alpha,0} \oplus \mathsf{L}_{\beta,0}) \oplus (\Lambda_\alpha \oplus \Lambda_\beta) \cdot \Delta = (\mathsf{L}_{\alpha,0} \oplus \Lambda_\alpha \cdot \Delta) \oplus (\mathsf{L}_{\beta,0} \oplus \Lambda_\beta \cdot \Delta) = \mathsf{L}_{\alpha,\Lambda_\alpha} \oplus \mathsf{L}_{\beta,\Lambda_\beta}$$

on the output wire γ. While the 4-to-2 GRR technique [193] is not compatible with free XOR, the earlier 4-to-3 GRR technique [192] keeps compatible with free XOR. Afterward, Zahur, Rosulek, and Evans [195] improved the GC construction to 2κ bits per AND gate while keeping the XOR gates for free. The main idea behind their construction is to break an AND gate into two half gates for which the evaluator knows one input (*i.e.*, a public value). We review the half-gate construction as follows:

- **Construction of GCs:** The garbler computes a garbled row for an AND gate $(\alpha, \beta, \gamma, \wedge)$ as below:

$$\begin{aligned} G_{\gamma,0} :=& \mathsf{H}(\mathsf{L}_{\alpha,0}, \gamma) \oplus \mathsf{H}(\mathsf{L}_{\alpha,1}, \gamma) \oplus \lambda_\beta \cdot \Delta, \\ G_{\gamma,1} :=& \mathsf{H}(\mathsf{L}_{\beta,0}, \gamma) \oplus \mathsf{H}(\mathsf{L}_{\beta,1}, \gamma) \oplus \mathsf{L}_{\alpha,0} \oplus \lambda_\alpha \cdot \Delta. \end{aligned}$$

  The garbler also computes the 0-label on the output wire γ as:

$$\mathsf{L}_{\gamma,0} := \mathsf{H}(\mathsf{L}_{\alpha,0}, \gamma) \oplus \mathsf{H}(\mathsf{L}_{\beta,0}, \gamma) \oplus (\lambda_\alpha \cdot \lambda_\beta \oplus \lambda_\gamma)\Delta.$$

- **Evaluation of circuits:** For any AND gate $(\alpha, \beta, \gamma, \wedge)$, given $(\Lambda_\alpha, \mathsf{L}_{\alpha,\Lambda_\alpha})$ and $(\Lambda_\beta, \mathsf{L}_{\beta,\Lambda_\beta})$, the evaluator can evaluate the label on the output wire γ as follows:

$$\begin{aligned} \mathsf{Eval}(\Lambda_\alpha, \Lambda_\beta, \mathsf{L}_{\alpha,\Lambda_\alpha}, \mathsf{L}_{\beta,\Lambda_\beta}) :=& \mathsf{H}(\mathsf{L}_{\alpha,\Lambda_\alpha}, \gamma) \oplus \mathsf{H}(\mathsf{L}_{\beta,\Lambda_\beta}, \gamma) \oplus \Lambda_\alpha \cdot G_{\gamma,0} \oplus \Lambda_\beta \cdot (G_{\gamma,1} \oplus \mathsf{L}_{\alpha,\Lambda_\alpha}) \\ =& \mathsf{H}(\mathsf{L}_{\alpha,0}, \gamma) \oplus \mathsf{H}(\mathsf{L}_{\beta,0}, \gamma) \oplus (\Lambda_\alpha \cdot \Lambda_\beta \oplus \Lambda_\alpha \cdot \lambda_\beta \oplus \Lambda_\beta \cdot \lambda_\alpha) \cdot \Delta \\ =& \mathsf{H}(\mathsf{L}_{\alpha,0}, \gamma) \oplus \mathsf{H}(\mathsf{L}_{\beta,0}, \gamma) \oplus ((\Lambda_\alpha \oplus \lambda_\alpha) \cdot (\Lambda_\beta \oplus \lambda_\beta) \oplus \lambda_\alpha \cdot \lambda_\beta) \cdot \Delta \\ =& \mathsf{H}(\mathsf{L}_{\alpha,0}, \gamma) \oplus \mathsf{H}(\mathsf{L}_{\beta,0}, \gamma) \oplus (\Lambda_\gamma \oplus \lambda_\alpha \cdot \lambda_\beta \oplus \lambda_\gamma) \cdot \Delta \\ =& \mathsf{L}_{\gamma,0} \oplus \Lambda_\gamma \cdot \Delta = \mathsf{L}_{\gamma,\Lambda_\gamma}. \end{aligned}$$

  If the garbler sets $\mathsf{lsb}(\Delta) = 1$, then $\mathsf{lsb}(\mathsf{L}_{w,\Lambda_w}) = \mathsf{lsb}(\mathsf{L}_{w,0} \oplus \Lambda_w \cdot \Delta) = \mathsf{lsb}(\mathsf{L}_{w,0}) \oplus \Lambda_w$ for every wire w. Thus, the garbler can send $\mathsf{lsb}(\mathsf{L}_{w,0})$ for the output wire w of each AND gate to the

Table 2. Comparison of concretely efficient garbling schemes

Garbling schemes	Size of one GC[a] (κ bits per gate)		Calls to function H per gate				Assumption
			Garbler		Evaluator		
	AND	XOR	AND	XOR	AND	XOR	
Unoptimized textbook Yao [6, 190]	8	8	4	4	2.5	2.5	PRF[b]
Yao with point-permute [1, 191]	4	4	4	4	1	1	PRF
4-to-3 GRR [192]	3	3	4	4	1	1	PRF
4-to-2 GRR [193]	2	2	4	4	1	1	PRF
Free XOR [194]	3	0	4	0	1	0	Circular CRHF
Flexible XOR [200]	2	$\{0,1,2\}$	4	$\{0,2,4\}$	1	$\{0,1,2\}$	Circular CRHF
Half gates [195]	2	0	4	0	2	0	Circular CRHF
Fast 4-to-2 GRR [201]	2	1	4	3	2	1.5	PRF
Slicing and dicing [199]	1.5	0	≤ 6	0	≤ 3	0	Circular CRHF

[a]For GC size, a small constant additive term (*i.e.*, 5 bits) is ignored for [199].
[b]We use PRF to denote pseudo-random function.

evaluator. Then, the evaluator can compute $\Lambda_\gamma = \mathsf{lsb}(\mathsf{L}_{\gamma,\Lambda_\gamma}) \oplus \mathsf{lsb}(\mathsf{L}_{\gamma,0})$. Actually, the communication of bit $\mathsf{lsb}(\mathsf{L}_{w,0})$ for each AND gate can be omitted, if we define a label L_{w,z_w} corresponding to the actual bit z_w instead of the public value Λ_w for every wire w and set $\lambda_w = \mathsf{lsb}(\mathsf{L}_{w,0})$, and thus $\Lambda_w = \mathsf{lsb}(\mathsf{L}_{w,z_w}) = \mathsf{lsb}(\mathsf{L}_{w,0} \oplus z_w \cdot \Delta) = \lambda_w \oplus z_w$.
For every circuit-output wire w, the garbler can send the wire mask $\lambda_w \in \{0,1\}$ to the evaluator, who can compute the output bit $z_w := \Lambda_w \oplus \lambda_w$.

For security, it is unnecessary to model H as a random oracle, and instead is sufficient to require that H satisfies the notion of circular correlated robustness hash function (circular CRHF) [196]. In this case, we can use a random permutation such as a fixed-key AES to implement CRHFs [197, 198]. Given the hardware-instruction support, the computational efficiency of GCs can be significantly improved [197, 198]. This makes the efficiency bottleneck for GCs become the size of garbled circuits.

Zahur *et al.* [195] proved a lower bound of 2κ bits per AND gate in a model of linear garbling, which models the labels as a whole. Recently, Rosulek and Roy [199] broke through the half gates' lower bound by introducing a new technique called slicing and dicing, while keeping fully compatible with the free-XOR technique. In particular, they improved the communication cost of GCs to $1.5\kappa + 5$ bits per AND gate, when the computation is slightly more than half gates. In terms of techniques, they slice the garbled labels into two halves, and introduce more linear combinations to increase the linear-algebraic dimension in which the garbling scheme can operate. Besides, they also add some random control bits into the construction of GCs, where the control bits determine the linear combinations of labels and garbled ciphertexts, and are outside of the linear garbling model. However, the state-of-the-art garbling scheme [199] is more complex and involves many linear-algebraic operations. It may be a challenging task to give a simple description of their garbling scheme, *i.e.*, describe the garbling scheme as the clean composition of some simpler components similar to the half-gate construction.

In Table 2, we summarize the communication and computation costs of the known efficient garbling schemes by following the comparison table shown in [199], where the flexible-XOR technique [200] and fast 4-to-2 GRR technique [201] are also compared. Our survey only considers the garbling schemes on Boolean circuits, which allows to obtain the minimal size of garbled circuits. We refer the reader to [202–205] for garbling arithmetic circuits.

4.1.2 Secure multi-party computation

In the multi-party setting, constant-round MPC has to deal with the case that multiple parties collude to cheat an honest party. Therefore, we cannot let only one party construct garbled circuits, and instead make all parties jointly construct a garbled circuit in a distributed manner. We use distributed garbling schemes to generate multi-party garbled circuits. The first distributed garbling scheme was introduced by Beaver, Micali, and Rogaway [1] in 1990. Based on the distributed garbling, they presented a constant-round MPC protocol in the dishonest-majority setting, but this protocol has a very low concrete efficiency. In the semi-honest setting, we focus on the case of all-but-one corruption (*i.e.*, $n-1$ out of n parties allow

to be corrupted). We describe several work that construct more efficient constant-round MPC protocols in the honest-majority malicious setting in Section 4.2.

Surprisingly, in the dishonest-majority setting, the BMR garbling was first optimized until 2016 using the free-XOR technique [206]. Based on the optimized BMR garbled circuits, they proposed an efficient constant-round MPC protocol with semi-honest security. In particular, their improved BMR garbled circuit [206] is defined as follows:

- Every party P_i with $i \in [1,n]$ has the following secret values:
  (a) A global key $\Delta_i \in \{0,1\}^\kappa$.
  (b) For each wire w in the circuit, a share of a mask bit $\lambda_w \in \{0,1\}$.
  (c) For each wire w, two garbled labels $\mathsf{L}^i_{w,0}, \mathsf{L}^i_{w,1} \in \{0,1\}^\kappa$ such that $\mathsf{L}^i_{w,0} \oplus \mathsf{L}^i_{w,1} = \Delta_i$.
- For each AND gate $(\alpha, \beta, \gamma, \wedge)$, for each $u, v \in \{0,1\}$, all parties jointly compute the following:

$$G^j_{\gamma,u,v} := \left(\bigoplus_{i\in[1,n]} \mathsf{H}(\mathsf{L}^i_{\alpha,u}, \mathsf{L}^i_{\beta,v}, \gamma \| j) \right) \oplus \mathsf{L}^j_{\gamma,0} \oplus \big((u \oplus \lambda_\alpha) \cdot (v \oplus \lambda_\beta) \oplus \lambda_\gamma\big) \cdot \Delta_j.$$

  The multi-party garbled circuit consists of $(G^1_{\gamma,u,v}, \ldots, G^n_{\gamma,u,v})$ for the output wire γ of each AND gate and each $u, v \in \{0,1\}$.

While the BMR garbled circuit is symmetric that allows every party to evaluate the circuit, Wang, Ranellucci and Katz [109] proposed an asymmetric distributed garbling which only allows one party (e.g., P_1) to evaluate the circuit. The WRK garbled circuit is defined as follows:

- Every party P_i holds the same secret values as in the BMR garbled circuits.
- For each gate $(\alpha, \beta, \gamma, T)$ and $u, v \in \{0,1\}$, let $r_{\gamma,u,v} = (u \oplus \lambda_\alpha) \cdot (v \oplus \lambda_\beta) \oplus \lambda_\gamma$, and $r^i_{\gamma,u,v}$ be the i-th share of $r_{\gamma,u,v}$. Every pair of parties (P_i, P_j) hold the additive shares $\mathsf{K}_i[r^j_{\gamma,u,v}]$ and $\mathsf{M}_i[r^j_{\gamma,u,v}]$ of $r^j_{\gamma,u,v} \cdot \Delta_i$, such that $\mathsf{K}_i[r^j_{\gamma,u,v}] \oplus \mathsf{M}_i[r^j_{\gamma,u,v}] = r^j_{\gamma,u,v} \cdot \Delta_i$.
- For each $i \neq 1$, for each AND gate $(\alpha, \beta, \gamma, \wedge)$ and $u, v \in \{0,1\}$, P_i computes the following:

$$\mathsf{H}(\mathsf{L}^i_{\alpha,u}, \mathsf{L}^i_{\beta,v}, \gamma) \oplus \left(\{\mathsf{M}_j[r^i_{\gamma,u,v}]\}_{j\neq i}, \mathsf{L}^i_{\gamma,u,v} \oplus \Big(\bigoplus_{j\neq i} \mathsf{K}_i[r^j_{\gamma,u,v}] \Big) \oplus r^i_{\gamma,u,v} \cdot \Delta_i \right),$$

  where the output length of H is $n\kappa$ bits, while its output length is κ bits in the BMR garbled circuit.

Recently, Yang *et al.* [110] partially used the half-gate technique to further reduce the size of the WRK garbled circuit from $4n|\mathcal{C}|\kappa$ bits to $(4n-6)|\mathcal{C}|\kappa$ bits. The size of the BMR garbled circuit is $4n|\mathcal{C}|\kappa$ bits, and is larger than the WRK garbled circuit. The MPC protocols based on BMR garbled circuits will have 1–2 less online rounds than those based on WRK garbled circuits, if all parties obtain the output. The constant-round MPC protocols based on distributed garbling achieve *optimal* online communication cost. The main task for improving constant-round MPC is to reduce the communication cost in the preprocessing phase, while keeping the computation fast. However, the known constant-round MPC protocols that are concretely efficient has $O(n^2)$ computation complexity in the online phase. For a large number of parties, this becomes expensive. In 2017, Ben-Efraim *et al.* [207] constructed a constant-round MPC protocol in the BMR framework, which achieves the computation complexity of $O(1)$ in the online phase. This was done using key-homomorphic pseudorandom functions that can be constructed under the DDH/LWE assumption. Their protocol in the online phase is more efficient than the state-of-the-art semi-honest MPC protocol [206] with $O(n^2)$ computation complexity when the number of parties is at least 100. However, their approach is *not* compatible with the free-XOR optimization [194], which will introduce a large overhead in the preprocessing phase. Recently, Ben-Efraim *et al.* [208] used an encryption scheme that is both key-homomorphic and message-homomorphic based on the LPN assumption to construct a BMR-like garbled circuit that is compatible with free-XOR. Their LPN-based technique can obtain faster online computation when $n \geq 100$, but requires a rather expensive preprocessing phase. If the number of honest parties is relaxed to $h = n/c$ for some constant $1 < c < n$, the preprocessing phase can be accelerated significantly using the techniques in [106, 109].

4.2 Maliciously secure protocols

In the malicious setting, we first consider the two-party case, and then discuss the multi-party case in the dishonest-majority and honest-majority settings, respectively.

4.2.1 Secure two-party computation

For constant-round 2PC protocols, before 2017, one popular approach for designing maliciously secure protocols is to use the "Cut-and-Choose" (C&C) technique. There are two different flavors to use such technique. The first one is the circuit-level C&C approach that was introduced by Lindell and Pinkas [209] and optimized in [210–224], where many garbled circuits are prepared, a random subset of them are opened and verified, and the remaining unchecked circuits are evaluated. In the single-execution setting where a circuit is computed at once on an input, ρ garbled circuits need to be prepared for statistical security $2^{-\rho}$ and the most efficient 2PC protocol in this setting is by Wang *et al.* [224]. In the amortized setting where the same circuit is evaluated multiple times on different inputs, only $O(\rho/\log\tau)$ garbled circuits need to be prepared for amortizing over τ executions, and the best-known 2PC protocol in this setting is by Rindal and Rosulek [221]. The second one is the gate-level C&C approach that was introduced by Nielsen and Orlandi [225] and called LEGO, where a lot of individual garbled AND gates are prepared, a random subset of them are opened and verified, and the remaining unchecked garbled gates are soldered to a garbled fault tolerant circuit using the XOR-homomorphic commitments. Subsequently, the LEGO protocol was optimized in [226–231]. Compared to the circuit-level C&C approach, the gate-level C&C approach has a lower asymptotic complexity $O(\rho/\log|\mathcal{C}|)$ and supports the function-independent preprocessing where both circuit and input are unknown (where such preprocessing is not supported by the circuit-level C&C approach), but is less efficient in the amortized setting and has also lower efficiency for some functions in the single-execution setting.

In 2017, the milestone work by Wang, Ranellucci and Katz [108] proposed the authenticated garbling approach to construct highly-efficient 2PC protocols, where a single "authenticated" garbled circuit is constructed and transmitted. Their approach works in the following framework:

(1) **Function-independent preprocessing phase:** The parties run the TinyOT-like protocol to generate random authenticated bit shares and authenticated AND triples based on the BDOZ-style IT-MACs, where authenticated shares can be produced by executing the OT extension protocol. An authenticated AND triple is defined as $([\![a]\!],[\![b]\!],[\![c]\!])$ with $a,b,c\in\{0,1\}$ and $c=a\wedge b$.
(2) **Function-dependent preprocessing phase:** In this phase in which the circuit is known, the parties generate an authenticated garbled circuit in a distributed way. In the process of generating authenticated garbled circuits, the key observation is that we can use the same global key for garbled circuits and authenticated shares, and thus the MAC tags and local keys involved in authenticated shares are naturally set as the shares to be used for constructing garbled circuits.
(3) **Online phase:** The party P_1 evaluates the circuit and obtains the output.

Wang *et al.* [108] proposed and adopted the WRK garbled circuit, and thus only one party P_1 can evaluate the circuit. Concurrently, a similar approach is proposed by Hazay, Scholl, and Soria-Vazquez [106] based on the BMR garbled circuit. Later, Katz *et al.* [107] significantly optimized the 2PC protocol by 1) applying the half-gate technique into distributed garbling, and 2) improving the communication and computation of the TinyOT-like protocol. The 2PC protocol [107] can generate a garbled circuit with $2\kappa+1$ bits per AND gate in the preprocessing phase, and performs the circuit authentication separately in a batch in the online phase. For now, the state-of-the-art approach for maliciously secure 2PC is to adopt the distributed garbling approach [106–108], and is significantly outperform both C&C approaches. An interesting future work is to further reduce the size of distributed garbled circuits by Katz *et al.* [107].

4.2.2 Secure multi-party computation

Dishonest majority. For constant-round MPC protocols tolerating all-but-one malicious corruption, several studies [232–234] adopt the cut-and-choose approach or the combination approach of BMR and

SPDZ to construct MPC protocols. However, their concrete efficiency is very low. In this setting, the best-known MPC protocols [106, 109–111] follow the distributed garbling framework [106, 108] based on TinyOT-like protocols. These MPC protocols have the same structure as that of 2PC protocols [107, 108], but need to execute a consistency-check procedure to check the consistency of shares or global keys among multiple executions. Recently, Poddar *et al.* [235] applied the constant-round MPC protocol [109] with malicious security to build a system called Senate that allows n parties to collaboratively run analytical SQL queries while keeping individual data private. The state-of-the-art constant-round MPC protocol with malicious security was proposed by Yang *et al.* [110], and can be used to further improve the performance of the above application. While the half-gate optimization is totally applied in the construction of distributed garbling in the two-party setting [107], this is only done partially in the multi-party setting [110]. It is a challenge to totally apply the half-gate technique (or even the recent slicing-and-dicing technique [199]) to multi-party garbled circuits.

Honest majority. In the honest-majority setting, constant-round MPC protocols can be constructed based on replicated secret sharing using less communication and computation (see, e.g, [236–239]). In the three-party setting with at most one malicious corruption, Mohassel *et al.* [239] proposed the currently most efficient 3PC protocol with three rounds by constructing a single Yao-style garbled circuit, where the maliciously secure 3PC protocol has the essentially same cost as the semi-honest Yao's 2PC protocol. Concurrently, Ishai *et al.* [238] constructed a two-round 3PC protocol while three garbled circuits need to be sent. In the four-party setting with at most one malicious corruption, the state-of-the-art protocol was proposed by Byali *et al.* [236], and has five rounds of communication and needs to send a single Yao-style garbled circuit. This protocol can achieve the stronger security property, *i.e.*, GOD. In the five-party setting with at most two malicious corruptions, Chandran *et al.* [237] presented the best-known MPC protocol with 8 rounds of communication. They adopted the BMR garbled circuit to prevent collusion, and proposed a attested OT primitive to make the whole MPC protocol only rely on symmetric-key primitives without the need of OT protocols. In terms of communication cost, their maliciously secure protocol requires 60% less communication than the semi-honest MPC protocol with dishonest majority [206], and its semi-honest variant needs 8× less communication. Their construction [237] can be also extended to n parties with the corruption threshold $t \leq \sqrt{n}$. Later, building upon the work [237], secure five-party computation (5PC) with fairness or GOD was also constructed in [240] with a small overhead over the 5PC protocol [237] satisfying security with abort.

5 Oblivious transfer and oblivious linear-function evaluation

In this section, we describe the recent development and techniques of oblivious transfer (OT) and its important variants (*i.e.*, random OT and correlated OT). Furthermore, we present the arithmetic generalization of OT called oblivious linear-function evaluation (OLE) and its key variant (*i.e.*, vector OLE). While OT is mainly used in MPC protocols for Boolean circuits, OLE is mainly applied in MPC protocols for arithmetic circuits. In this survey, we mainly review the state-of-the-art techniques to construct (correlated) OT, and give a concise overview of the techniques to design (vector) OLE. Note that OLE has the same importance as OT. Additionally, vector OLE can be designed in the same framework as correlated OT for the state-of-art technique based on learning parity with noise (LPN). We note that homomorphic encryption (HE) is a key technique to generate (vector) OLE correlations, although it is not described in detail in this section. The recent techniques based on LPN variants allow to obtain sublinear communication complexity, compared to linear communication complexity based on HE.

5.1 Oblivious transfer

Oblivious transfer (OT) [95, 96] is a fundamental cryptographic primitive between a sender S and a receiver R, which enables R to obtain only one of the two input messages of S, while S learns nothing on the R's choice bit. It can be used to construct not only the Yao's 2PC protocol but also a lot of other MPC protocols with both semi-honest and malicious security. In addition, OT can also be used to design a lot of cryptographic protocols of other kinds. OT protocols can be constructed from different cryptographic assumptions, including decisional Diffie–Hellman (DDH) [241–245], computational Diffie–Hellman (CDH) [244, 246–248], learning with errors (LWE) [242, 245, 249–251], learning parity with

noise (LPN) [247] and commutative supersingular isogeny Diffie–Hellman (CSIDH) [252]. However, when a large number of OT correlations need to be generated (particularly for MPC applications), these OT protocols based on public key operations are very expensive. To deal with this problem, Beaver [253] introduced the notion of OT extension, in which a small number of base OTs are extended efficiently to a large number of OTs (even any polynomial number of OTs) using fast operations. The first OT extension protocol by Beaver [253] uses the pseudorandom generator (PRG) in a non-black-box way, and thus is only theoretically interesting. For now, concretely efficient OT extension protocols are divided into two styles: one based on the IKNP framework [130] and the other in the PCG framework [162, 254]. While the IKNP-style protocols adopt the symmetric-key primitive PRG to perform extension and support *chosen* choice bits, the PCG-style protocols utilize the sparse feature of the noise in the LPN problem [255] to realize extension and only allow random choice bits.[2] OT extension of both styles adopts the following structure from weak OTs to standard OTs:

$$\text{Correlated OT (COT)} \Rightarrow \text{Random OT (ROT)} \Rightarrow \text{OT},$$

where COT requires two messages (m_0, m_1) of the sender satisfying the fixed correlation (*i.e.*, $m_0 \oplus m_1 = \Delta$ for a fixed string Δ), ROT only allows to output two uniformly random messages, and OT allows to input arbitrary two messages. Both transformations (*i.e.*, COT $\Rightarrow$ ROT and ROT $\Rightarrow$ OT) are standard and recalled as follows:

- **COT $\Rightarrow$ ROT:** Given a CRHF $\mathsf{H} : \{0,1\}^{2\kappa} \to \{0,1\}^{\kappa}$ and a COT correlation $(\mathsf{K}[b], (b, \mathsf{M}[b]))$ with $\mathsf{M}[b] = \mathsf{K}[b] \oplus b \cdot \Delta$, where $\mathsf{K}[b], \Delta \in \{0,1\}^{\kappa}$ are two random strings held by the sender and $b \in \{0,1\}, \mathsf{M}[b] \in \{0,1\}^{\kappa}$ are held by the receiver, a ROT correlation $((r_0, r_1), (b, r_b))$ can be computed without any interaction as below:

  $$r_0 := \mathsf{H}(\mathsf{K}[b], i), \quad r_1 := \mathsf{H}(\mathsf{K}[b] \oplus \Delta, i), \quad r_b := \mathsf{H}(\mathsf{M}[b], i),$$

  where i is an index associated with the COT correlation. While the index i can be omitted in the semi-honest setting, it is necessary for malicious security [198].
- **ROT $\Rightarrow$ OT:** Given a ROT correlation $((r_0, r_1), (b, r_b))$ where the sender obtains two random strings r_0, r_1 and the receiver gets a choice bit b and the string r_b, a standard OT correlation $((m_0, m_1), (b, m_b))$ can be constructed using the "one-time padding" technique as follows:
  - The sender sends $\tau_0 = m_0 \oplus r_0$ and $\tau_1 = m_1 \oplus r_1$ to the receiver, who computes $m_b = \tau_b \oplus r_b$.

Therefore, we can focus on designing concretely efficient COT protocols, and then transform them to standard OT protocols. In addition, COT protocols are able to be used to generate BDOZ-style authenticated shares using the TinyOT-like protocols [104, 106–110] as well as SPDZ-style authenticated shares using the bit-decomposition idea [43, 150]. For GCs with free-XOR, the garbled labels for every wire in the circuit satisfy the COT correlation, and thus can be transmitted obliviously from a garbler to a evaluator using a COT protocol, *i.e.*, COT can also be straightforwardly used in MPC protocols.

The semi-honest IKNP protocol [130] (improved in [113]) works roughly as follows: 1) execute a base-OT protocol (relying on public-key operations) to generate κ ROT correlations in the setup phase, by switching the role of the sender and receiver and then 2) extend κ ROT correlations to a large number of COT correlations in the extension phase, using PRG and switching column vectors into row vectors. The extended phase can be executed iteratively to generate an unlimited number of COT correlations, when using the same setup phase [113]. Later, the IKNP-style OT extension protocols with malicious security were proposed in [135, 136]. Using the random-linear-combination approach, the maliciously secure protocol by Keller, Orsini, and Scholl [136] achieves the best efficiency in the IKNP framework, and has the communication cost matching that of the best-known semi-honest protocol [113]. While the IKNP-style OT extension protocols enjoy fast computation, they require linear communication cost (*i.e.*, κ bits per COT correlation).

Another style of OT extension protocols lies in the pseudorandom correlation generator (PCG) framework [162, 254].[3] In general, the PCG-style OT extension protocols [132, 134, 162, 256] are able to generate

[2] We note that OTs with random choice bits can be transformed into OTs with chosen choice bits at the cost of additionally communicating 1 bit per OT, using the precomputing OT technique [188].

[3] PCG allows two parties to generate two *short* correlated seeds in the setup phase and then use the seeds to produce a *long* correlated randomness such as COT correlations without any communication.

COT correlations with sublinear communication (*i.e.*, $\tilde{O}(\sqrt{N})$ for producing N COT correlations), but need more computation than IKNP-style protocols. To simplify the following description, we now give an informal definition of COT in a vector form. Specifically, sender S obtains a uniform global key $\Delta \in \mathbb{F}_{2^\kappa}$ and a random vector $\mathbf{v} \in \mathbb{F}_{2^\kappa}^N$, while receiver R holds a uniform choice-bit vector $\mathbf{u} \in \mathbb{F}_2^N$ and a vector $\mathbf{w} \in \mathbb{F}_{2^\kappa}^N$ such that $\mathbf{w} = \mathbf{v} + \Delta \cdot \mathbf{u}$.

For both semi-honest and malicious security, the state-of-the-art PCG-style COT protocols [132, 134] are constructed in the following three layers:

(1) **SPCOT:** Construct a single-point COT (SPCOT) protocol, a variant of COT where the Hamming weight of the choice-bit vector $\mathbf{u}$ is exactly 1 (*i.e.*, $\mathsf{HW}(\mathbf{u}) = 1$). We can use a point $\alpha \in [1, N]$ to represent the location of the single non-zero entry, meaning that $u_\alpha = 1$ and $u_i = 0$ for $i \neq \alpha$. We can construct the SPCOT protocol using the following approach:

- **Semi-honest security:** The best-known SPCOT protocol [132] in the semi-honest setting adopts the designing idea of the puncturable pseudorandom function (PPRF) construction based on the GGM tree [257].[4] In particular, a PPRF is a special pseudorandom function F, which can generate a normal key k and a punctured key $\mathsf{k}\{\alpha\}$ for an input α, such that k can be used to evaluate F at each point, and $\mathsf{k}\{\alpha\}$ allows to evaluate F at every point except for α without leaking any information about $F(\mathsf{k}, \alpha)$ [258, 259].
  Using the binary-tree structure of GGM-PPRF, sender S can transmit $\mathsf{k}\{\alpha\}$ to the receiver R without knowing any information on α, by executing the OT protocol $\log N$ times in parallel. We refer the reader to [132, 260] for details. Using key k, S can compute vector $\mathbf{v}$ as $v_i := F(\mathsf{k}, i) \in \mathbb{F}_{2^\kappa}$ for $i \in [1, N]$. With $\mathsf{k}\{\alpha\}$, R is able to compute $w_i := F(\mathsf{k}, i) \in \mathbb{F}_{2^\kappa}$ for $i \neq \alpha$. To define value w_α, S could send $\tau = \Delta + \sum_{i\in[1,N]} v_i \in \mathbb{F}_{2^\kappa}$ to R, who can compute $w_\alpha := \tau + \sum_{i\neq\alpha} w_i$. Since $w_i = v_i$ for each $i \in [1, N]$, $i \neq \alpha$, we have that $w_\alpha = \tau + \sum_{i\neq\alpha} w_i = \Delta + \sum_{i\in[1,N]} v_i + \sum_{i\neq\alpha} v_i = v_\alpha + \Delta$, where addition is performed over binary field $\mathbb{F}_{2^\kappa}$. Therefore, we obtain that $\mathbf{w} = \mathbf{v} + \Delta \cdot \mathbf{u}$ holds, where R defines $\mathbf{u}$ as $u_\alpha = 1$ and $u_i = 0$ for $i \neq \alpha$.
- **Malicious security:** The above SPCOT protocol allows a malicious sender S to send incorrect messages in the OT protocol executions, so that the punctured key obtained by receiver R does *not* correspond to the punctured point α. The deviation of the outputs of two parties can be detected by the receiver by executing a consistency-check procedure. The high-level idea for the state-of-the-art consistency check [134] is as follows:
  (a) From $\mathbf{v} + \mathbf{w} = \Delta \cdot \mathbf{u}$, we apply a random linear combination defined by uniformly random coefficients $\chi_1, \ldots, \chi_N \in \mathbb{F}_{2^\kappa}$ sampled by R into two sides of the equation. According to $u_\alpha = 1$ and $u_i = 0$ for $i \neq \alpha$, we obtain the following result:

$$\sum_{i\in[1,N]} \chi_i \cdot v_i + \sum_{i\in[1,N]} \chi_i \cdot w_i = \chi_\alpha \cdot \Delta.$$

  (b) Using the approach underlying the MASCOT protocol [43], S and R can compute the additive shares of $\chi_\alpha \cdot \Delta$:

$$Y + Z = \chi_\alpha \cdot \Delta,$$

  where it needs extra κ COT correlations.
  (c) Combining two equations, we have the following:

$$\begin{aligned} V &:= \sum_{i\in[1,N]} \chi_i \cdot v_i + Y \\ &= \sum_{i\in[1,N]} \chi_i \cdot w_i + Z := W. \end{aligned}$$

  The remaining task is to check $V = W$ by running an equality-test protocol. Since V and W are unnecessary to be kept secret, the equality-test protocol can be constructed in a highly efficient manner using a cryptographic hash function [69, 134].

[4] Concurrently, Schoppmann *et al.* [260] used the same approach to design an arithmetic variant of SPCOT.

(2) **MPCOT:** Construct a multi-point COT (MPCOT) protocol, a variant of COT with $\mathsf{HW}(\mathbf{u}) = t$ for a parameter $t > 1$. Based on the SPCOT protocol, an MPCOT protocol can be constructed in a fairly straightforward way:
- Given the length N of MPCOT vectors, two parties S and R can execute the SPCOT protocol t times with each the outputting length N/t. Then, for $i \in [1, t]$, S obtains $\mathbf{v}_i \in \mathbb{F}_{2^\kappa}^{N/t}$, and R gets $\mathbf{w}_i \in \mathbb{F}_{2^\kappa}^{N/t}$ and $\mathbf{u}_i \in \mathbb{F}_2^{N/t}$ with $\mathsf{HW}(\mathbf{u}_i) = 1$. Sender S defines $\mathbf{v} := (\mathbf{v}_1, \ldots, \mathbf{v}_t) \in \mathbb{F}_{2^\kappa}^N$, and receiver R sets $\mathbf{w} := (\mathbf{w}_1, \ldots, \mathbf{w}_t) \in \mathbb{F}_{2^\kappa}^N$ and $\mathbf{u} := (\mathbf{u}_1, \ldots, \mathbf{u}_t) \in \mathbb{F}_2^N$ where $\mathsf{HW}(\mathbf{u}) = \sum_{i \in [1,t]} \mathsf{HW}(\mathbf{u}_i) = t$.
- The MPCOT protocol described as above needs $t \log N/t$ OT correlations to execute the SPCOT sub-protocol. These OT correlations (thousands of OT correlations for concrete parameters t, N [132, 134]) can be generated using the IKNP-style OT extension protocol [113, 136]. For malicious security, extra $t\kappa$ COT correlations are required. This can be optimized to κ COT correlations by combining t consistency checks into a single check (see [134] for details).
- In the malicious setting, a malicious sender S may use different Δ in the t SPCOT protocol executions. Thus, we need a consistency-check procedure to guarantee the consistency of Δ. The state-of-the-art consistency check [134] guarantees the consistency of Δ *for free*, as the SPCOT correlations are always assured to use the same Δ as the extra κ COT correlations, which have guaranteed the consistency of Δ.

(3) **COT from LPN:** This procedure extends MPCOT correlations to COT correlations with uniform choice bits, based on the LPN assumption. For both semi-honest and malicious security, this procedure is the same, and only involves local computation.
- Based on the MPCOT protocol, two parties S and R can generate a length-N MPCOT vector $(\mathbf{s}, (\mathbf{r}, \mathbf{e}))$ such that $\mathbf{r} = \mathbf{s} + \Delta \cdot \mathbf{e} \in \mathbb{F}_{2^\kappa}^N$. Here, we can view $\mathbf{e} \in \mathbb{F}_2^N$ as the noise vector of an LPN problem, such that $\mathbf{e}$ is divided into t consecutive sub-vectors of length N/t where each sub-vector has a single non-zero entry at a random position. Such distribution is called as a *regular* noise distribution. As analyzed and observed in previous work [114, 132, 162, 254, 261], no known attack exploits a regular noise distribution, and performs significantly better than a uniform noise distribution where $\mathbf{e}$ is a uniform vector such that $\mathsf{HW}(\mathbf{e}) = t$.
- Based on LPN assumptions of two different flavors, S and R can generate COT correlations in the following two ways:
  (a) **Dual LPN:** Informally, the dual-LPN assumption with a regular noise distribution $\mathcal{D}_t$ states that:
  $$\mathbf{e} \cdot \mathbf{H} \overset{\text{c}}{\approx} \mathbf{u},$$
  where $\mathbf{e} \leftarrow \mathcal{D}_t$, $\mathbf{H} \in \mathbb{F}_2^{N \times n}$ is a matrix created by a code generation algorithm, $\mathbf{u} \in \mathbb{F}_2^n$ is a uniform vector and $N = c \cdot n$ for a compression parameter $c > 1$ (*e.g.*, $c = 2$ or 4). The dual-LPN assumption is also known as the regular syndrome decoding (RSD) assumption, which is introduced in [261] as the assumption underlying the security of the candidate fast syndrome-based (FSB) hash function for the SHA-3 competition.
  Given an MPCOT vector $(\mathbf{s}, (\mathbf{r}, \mathbf{e}))$, S and R output a COT vector $(\mathbf{v}, (\mathbf{u}, \mathbf{w}))$ as follows:
  $$\mathbf{v} = \mathbf{s} \cdot \mathbf{H} \in \mathbb{F}_{2^\kappa}^n, \quad \mathbf{u} = \mathbf{e} \cdot \mathbf{H} \in \mathbb{F}_2^n, \quad \mathbf{w} = \mathbf{r} \cdot \mathbf{H} \in \mathbb{F}_{2^\kappa}^n.$$
  (b) **Primal LPN:** Informally, the primal-LPN assumption with a regular noise distribution $\mathcal{D}_t$ states that:
  $$\mathbf{a} \cdot \mathbf{A} + \mathbf{e} \overset{\text{c}}{\approx} \mathbf{u},$$
  where $\mathbf{a} \leftarrow \mathbb{F}_2^k$ is a uniform vector, $\mathbf{A} \in \mathbb{F}_2^{k \times n}$ is a matrix created by a code generation algorithm, $\mathbf{e} \leftarrow \mathcal{D}_t$, $\mathbf{u} \in \mathbb{F}_2^n$ is a uniform vector and $k < n$.
  In this case, two parties S and R additionally need a length-k COT vector $(\mathbf{b}, (\mathbf{a}, \mathbf{c}))$ such that $\mathbf{c} = \mathbf{b} + \Delta \cdot \mathbf{a} \in \mathbb{F}_{2^\kappa}^k$ and $\mathbf{a} \in \mathbb{F}_2^k$ is a uniform vector. Then, given an MPCOT vector $(\mathbf{s}, (\mathbf{r}, \mathbf{e}))$, S and R can output vector $\mathbf{v}$ and two vectors $(\mathbf{u}, \mathbf{w})$, respectively, as follows:
  $$\mathbf{v} = \mathbf{b} \cdot \mathbf{A} + \mathbf{s} \in \mathbb{F}_{2^\kappa}^n, \quad \mathbf{u} = \mathbf{a} \cdot \mathbf{A} + \mathbf{e} \in \mathbb{F}_2^n, \quad \mathbf{w} = \mathbf{c} \cdot \mathbf{A} + \mathbf{r} \in \mathbb{F}_{2^\kappa}^n.$$

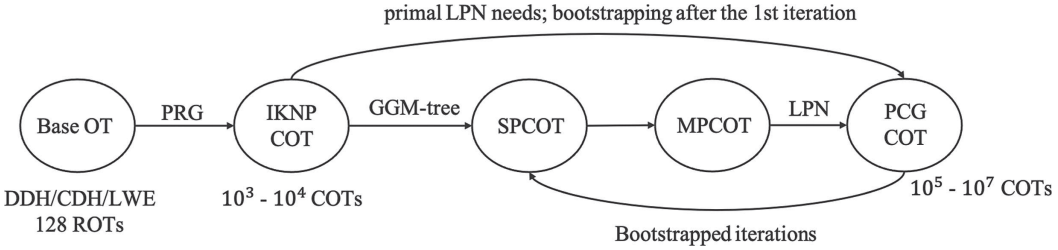

Figure 4. Structure of the PCG-based COT protocols. When $\kappa = 128$, we need to use public-key operations to compute 128 base ROT correlations based on the DDH, CDH, or LWE assumptions. We can use the IKNP-style OT extension protocol to generate COT correlations in an order of magnitude $10^3 - 10^4$, and then extend them to COT correlations in an order of magnitude $10^5 - 10^7$ (or even more) based on the LPN assumption

In general, the COT protocol based on dual LPN enjoys a lower communication but has a slower computation, while that based on primal LPN enjoys a faster computation but has a higher communication.

- **Bootstrapped iterations:** To generate an unlimited number of COT correlations, two parties can execute the COT protocol *iteratively* in a *bootstrapped* mode. In particular, let M be the number of all setup COT correlations to execute the whole COT protocol, where $M = t \log \frac{N}{t}$ for dual LPN and $M = k + t \log \frac{n}{t}$ for primal LPN in the semi-honest setting, and M is further increased by κ for malicious security. Every iteration produces n COT correlations using the setup COT correlations, and outputs $n - M$ COT correlations where the remaining M COT correlations are stored and bootstrapped as the refreshed setup COT correlations to be used in the next iteration. In the first iteration, M setup COT correlations can be generated using the IKNP-style OT extension protocol [113, 136]. When a huge number of COT correlations are required and many iterations are executed, the setup cost for generating setup COT correlations in the first iteration can be amortized to negligible.

In Figure 4, we present the structure of the PCG-based COT protocol described as above. According to the best-known implementations, the IKNP-style protocols are highly efficient to compute thousands of COT correlations, and the PCG-style protocols will be more efficient if millions of COT correlations are required even if the network bandwidth is large enough.

Recently, Rindal *et al.* [133] proposed a new variant of the dual-LPN assumption, using a structured and sparse matrix $\mathbf{H}$ generated a new LDPC code. When applying the new dual-LPN problem into the semi-honest COT protocol by Boyle *et al.* [132], they showed that the COT protocol based on dual LPN can simultaneously achieve lower communication and faster computation than the best-known COT protocol based on primal LPN [134], as the new structured LPDC codes [133] support fast encoding operation. Based on the new dual-LPN assumption [133], the COT protocol [132] could even obtain 37% less computation than the best-known IKNP-style protocol [113]. However, the efficiency gain builds upon an aggressive dual-LPN problem based on heuristically designed linear codes. Rindal *et al.* [133] analyzed two key properties of the underlying linear codes (including large minimum distance) under the linear test framework to establish a degree of confidence about the hardness of the new dual-LPN problem. More analyses on the new dual-LPN problem are encouraged to establish more confidence on the security of their COT protocol. When applying the state-of-the-art consistency check by Yang *et al.* [134] into the semi-honest COT protocol [132] based on the new dual-LPN assumption [133], we can obtain the currently most efficient COT protocol with malicious security. This consistency check along with the check technique by Boyle *et al.* [132] allows the malicious sender to guess some positions of non-zero entries of noise vector $\mathbf{e}$ in a selective failure manner, *i.e.*, an incorrect guess will be caught. This means that the adversary is allowed to query (on average) one-bit information on the noise vector. In this case, it is worth analyzing whether the current parameter selection of the underlying LPN assumptions has already been sufficient to achieve the κ-bit security level (e.g., $\kappa = 128$).

Recently, Boyle *et al.* [256] proposed the notion of pseudorandom correlation function (PCF), and gave an efficient PCF construction for generating COT correlations under a variable-density variant of the LPN assumption (VDLPN). While PCG only allows to generate a fixed length of correlated randomness (e.g., COT) in an all at once way and does not support the stateful incremental evaluation enabled by PRG

in a "stream-cipher" mode, PCF can produce correlated randomness on-the-fly and offer the ability to securely generate virtually unbounded number of correlated randomness. In particular, PCF allows two parties to generate two short correlated keys k_0 and k_1 in the setup phase, and then use the keys to compute COT correlations on-the-fly, *i.e.*, $v_i = \mathsf{PCF}(\mathsf{k}_0, x_i)$ and $(u_i, w_i) = \mathsf{PCF}(\mathsf{k}_1, x_i)$ for a uniform string x_i such that $w_i = v_i + \Delta \cdot u_i$ where Δ can be involved in key k_0. Boyle *et al.* [256] only presented the semi-honest construction to distribute the short keys for computing COT correlations, which may have an efficiency advantage than the PCG approach when the number N of resulting COT correlations is very huge (e.g., $N \approx 2^{48}$ or even larger).

5.2 Oblivious linear-function evaluation

OLE. Oblivious linear-function evaluation (OLE) is an arithmetic generalization of OT, and is particularly useful for designing MPC protocols for arithmetic circuits over large fields [120, 146, 262–264]. In particular, OLE directly gives a two-party additive sharing of the multiplication of two secret values. Therefore, by a pairwise OLE protocol execution, we can use OLE to generate Beaver multiplication triples without authentication in the multi-party setting. OLE can be constructed using OT extension and Gilboa multiplication approach [43, 150], and has a cheap computation cost but a much high communication cost. There exists a standard approach to design OLE using additively homomorphic encryption (AHE) based on RLWE, which has been used in Overdrive [152] and the recent work [265], where a receiver R sends $\mathsf{Enc}(x)$ to sender S, and then S computes $\mathsf{Enc}(y) = u \cdot \mathsf{Enc}(x) + v$ and sends it to R who decrypts to obtain $y = u \cdot x + v \in \mathbb{F}$ for a large field $\mathbb{F}$. Here, the AHE needs to satisfy the circuit privacy property. In addition, OLE can also be constructed from somewhat homomorphic encryption [103, 149], but will require a larger communication. Without relying on homomorphic encryption, OLE is also able to be built directly from Ring-LWE [266, 267]. Besides, we can also construct OLE protocols from OT and noisy Reed-Solomon encodings [97, 264, 268], or Paillier encryption [269]. Among all the OLE protocols, the protocols [152, 265] based on AHE obtain the best communication efficiency, and the protocol [266] from RLWE has the optimal one round of communication.

Recently, Boyle *et al.* [162] proposed an OLE construction based directly on LPN, which has very lower communication cost than the above OLE protocols but needs the computational cost of at least $O(N^2)$ for generating N OLE correlations. Later, they [161] solved the computational problem using a variant of the ring-LPN assumption, and constructed an OLE protocol for computing a large number of OLE correlations. This OLE protocol has very lower communication cost than the protocols based on RLWE, and provides a computational complexity of $\tilde{O}(N)$. Their PCG approach based on ring-LPN is a nice approach to generate a large number of OLE correlations (e.g., $N = 2^{20}$). For a small number of OLE correlations, the approaches based on RLWE may be better. Based on ring-LPN, the resulting OLE correlations are random (*i.e.*, $u, v, x \in \mathbb{F}$ are uniformly random), but are sufficient to design MPC protocols where only random multiplication triples need to be generated in the preprocessing phase.

VOLE. Vector oblivious linear-function evaluation (VOLE) is an arithmetic generalization of COT to a large field and defined as follows:

- A sender holds a uniform global key $\Delta \in \mathbb{F}$.
- For each VOLE execution, the sender obtains a vector $\mathbf{v} \in \mathbb{F}^N$, and a receiver gets two vectors $\mathbf{w}, \mathbf{u} \in \mathbb{F}^N$, such that $\mathbf{w} = \mathbf{v} + \Delta \cdot \mathbf{u}$.

We have a standard transformation from COT to OT using CRHFs [130]. This is not the case for VOLE and OLE, as the underlying field $\mathbb{F}$ is large and the sender cannot enumerate all possible values w.r.t. $x \in \mathbb{F}$. Similar to OLE, VOLE can be built based on OT extension [43] or AHE [152, 265], where the latter has a lower communication. The VOLE protocols [152, 265] based on AHE have the communication complexity linear to the output length of VOLE. Based on the LPN assumption, the PCG approach [254] can construct VOLE protocols with sublinear communication, and is the most promising approach to produce a large number of VOLE correlations (*e.g.*, $N \geq 10^5$). Subsequently, this approach was further optimized in [69, 132–134, 162, 256, 260]. The efficiency and security comparisons among these VOLE protocols based on LPN is similar to the COT case shown in the previous subsection.

The state-of-the-art VOLE protocols [69, 133] adopt the same framework as the best-known COT protocols [133, 134] based on dual-LPN or primal-LPN, except that an additional VOLE correlation needs to be generated in a single-point VOLE protocol execution as the single non-zero element is uniform in large field $\mathbb{F}$ rather than equal to 1. Additionally, for VOLE, we need to use the LPN assumption over a large field $\mathbb{F}$ instead of $\mathbb{F}_2$. We are able to use the VOLE protocols [152, 265] based on AHE to generate the VOLE correlations in the setup phase. Besides, we can use the PCF approach to generate VOLE correlations under the VDLPN assumption [256], and may have an efficiency advantage than the PCG approach if the number of resulting VOLE correlations is very huge. Similar to the case of COT, we can use the state-of-the-art consistency check [69, 134] to construct maliciously secure VOLE protocols.

6 MPC application to machine learning

Recent advances in machine learning (ML) have driven a lot of real-life applications, such as healthcare, financial risk analysis, facial recognition, image and video analysis for self-driving cars, recommendation systems, text translation, voice assistants, image classification, *etc.* The level of accuracy as required is high for mission-critical applications (*e.g.*, healthcare). Accuracy is mainly governed by two factors: 1) the large amount of computing power that is demanded to train deep learning models; 2) the variance in datasets, which comes from collecting data from multiple diverse sources and is generally infeasible for a single company to achieve.

Toward this, multiple companies (e.g., Microsoft, Amazon, Google) provide with machine learning as a service (MLaaS), which works in the following two different ways:

- **Inference:** A company offers a trained ML model, and a customer is able to query a feature input to obtain the inference result.
- **Training:** Multiple companies work together to train a high accuracy model using their datasets.

In the first scenario, companies want to keep the ML model secret as it may take a lot of money to train a model, and customers wish to protect the privacy of their inputs where the input information may be sensitive such as personal health data or faces. In the second scenario, companies would not be willing to share their data since data are proprietary information of a company and these companies may be competitive, and are prohibited from sharing client information due to privacy laws. Here, we say that an ML model is kept secret, meaning that the model parameters are hidden, but the model structure (e.g., which functions are used) is still known. It is a challenge to protect the privacy of model structure while keeping PPML concretely efficient.

Therefore, to address the above privacy concerns in ML applications, privacy-preserving machine learning (PPML) is highly desirable, and has emerged as a flourishing research area. In particular, PPML allows ML computations over private data, while ensuring the privacy of the data. Due to the privacy-protection requirement, PPML makes the already compute-intensive ML algorithms more demanding in terms of high computation power and large communication cost. However, many everyday users have no such computation and communication capacities to execute PPML. Thus, it may be economical and convenient for users to securely outsource an ML task to a set of powerful and specialized cloud servers in a pay-per-use manner, where the security is guaranteed if at most t servers of n servers collude to cheat (either $t < n$ or $t < n/2$ depending on the concrete MPC protocols used). In this case, the inference and training can be realized in the following way:

- **Outsourcing inference:** A company may host its trained ML model in a secret-shared way to n (untrusted) servers. A customer can secretly share its feature input among the same n servers. The servers can compute an inference result in a shared fashion and return the result to the customer.
- **Outsourcing training:** Multiple companies can secretly share their datasets to a set of (untrusted) servers, who cooperatively train a common model on their joint datasets while keeping their individual dataset private.

MPC is one of key techniques to realize PPML, and is the most promising approach to perform PPML in the above outsourced computation setting based on secret sharings. A series of PPML protocols have been built upon MPC techniques. We can partition these PPML protocols into two categories:

Table 3. Comparison of various PPML protocols

	PPML	Capability		Threat Model		Techniques	Neural
		Inference	Training	Semi-honest	Malicious		Networks
2PC	SecureML [24]	√	√	√		HE/GC/ASS	From [24]
2PC	MiniONN [277]	√		√		HE/GC/ASS	From [24, 277]
2PC	GAZELLE [20]	√		√		HE/GC/ASS	From [24, 277]
2PC	EzPC [278]	√		√		GC/ASS	From [24, 277]
2PC	XONN [29]	√		√		GC/ASS	VGG-16 [279]
2PC	QUOTIENT [18]	√	√	√		OT/GC/ASS	From [18]
2PC	MP2ML [280]	√		√		HE/GC/ASS	CryptoNets [281]
2PC	CrypTFlow2 [28]	√		√		HE/OT/ASS	DenseNet-121 [282]
2PC	Delphi [22]	√		√		HE/GC/ASS	VGG-16 [279]
2PC	QuantizedNN [283]	√		√	Abort	HE/OT/ASS	MobileNets [284]
2PC	SIRNN [27]	√		√		OT/ASS	Heads [285]
3PC	Chameleon [286]	√		√		GC/ASS	AlexNet [287]
3PC	ABY^3 [23]	√	√	√		GC/ASS	From [24, 277]
3PC	ASTRA [288]	√	√	√	Abort	ASS/RSS	From [24]
3PC	SecureNN [289]	√	√	√		ASS	From [24, 277]
3PC	BLAZE [26]	√	√	√	Fairness	ASS/RSS	From [24]
3PC	QuantizedNN [283]	√		√	Abort	RSS	MobileNets [284]
3PC	CrypTFlow [21]	√		√		ASS	DenseNet-121 [282]
3PC	SWIFT [290]	√	√	√	GOD	ASS/RSS	VGG-16 [279]
3PC	CryptGPU [31]	√	√	√		RSS	ResNet-152 [291]
3PC	Falcon [292]	√	√	√	Abort	RSS	VGG-16 [279]
4PC	FLASH [293]	√	√	√	GOD	ASS/RSS	From [24]
4PC	SWIFT [290]	√	√	√	GOD	ASS/RSS	VGG-16 [279]
4PC	Trident [19]	√	√	√	Fairness	GC/ASS/RSS	From [24]
4PC	Tetrad [294]	√	√	√	GOD	GC/ASS/RSS	VGG-16 [279]

Note: All protocols for secure three-party/four-party computation (*i.e.*, 3PC/4PC) tolerate one corruption, and thus belong to the honest-majority setting. For malicious adversaries, "Abort", "Fairness", and "GOD" denote the PPML protocols that achieve security with abort, fairness, and guaranteed output delivery, respectively. For the underlying LSSS, we use "ASS" and "RSS" to denote the additive secret sharing and the replicated secret sharing, respectively. If a PPML protocol supports multiple neural-network architectures, we only describe the one with largest parameters for private ML inference.

one is in the dishonest-majority setting, and the other is in the honest-majority setting. We surveyed the known PPML protocols based on MPC, and compare them in Table 3. All PPML protocols shown in Table 3 along with other PPML protocols [30, 34, 270–276] are customized in the following ways:

- Based on the known MPC protocols, improve the ML algorithms to make them more MPC-friendly.
- According to the definitions of ML algorithms, tailor the known MPC protocols.

These customized PPML protocols can obtain high efficiency for specific learning tasks. Recently, Zheng *et al.* [35] designed a platform for privacy-preserving training and inference of generic ML tasks, which supports new neural-network architectures but has a lower efficiency.

In the dishonest-majority setting, the PPML protocols focus on the two-party case, except for two protocols Helen [36] and Cerebro [35]. Both Helen and Cerebro implemented the inference and training of ML algorithms among 4 parties and 2–12 parties, respectively, and allow the adversary to be semi-honest or malicious. For 12 parties tolerating 11 semi-honest corruptions, the recent PPML protocol Cerebro [35] can perform an inference of the decision tree with 12 layers in average time about 20 s. In the semi-honest setting with six parties, Cerebro can implement the logistic regression training in about 16 minutes, and the linear regression training in about 100 s. According to the experimental results in Cerebro [35], the maliciously secure PPML protocol is 61–3300× slower than the semi-honest version. In the multi-party setting with dishonest majority, the model for ML inference is small, and the dataset and neural-network architecture for ML training is also small. More efficiency optimizations need to be exploited to support larger datasets and models. The maliciously secure protocols need to be further improved to reduce the overhead over the semi-honest protocols. Now, we turn our attention to the two-party case. Most of the two-party PPML protocols consider the semi-honest adversaries. The only

exception is QuantizedNN [283], which uses SPD$\mathbb{Z}_{2^k}$ [157, 158] and Overdrive [152] to design maliciously secure protocols, where SPD$\mathbb{Z}_{2^k}$ and Overdrive have been implemented in the MP-SPDZ library [17]. Their maliciously secure protocol [283] is roughly 3–9× slower than the semi-honest protocol. In the semi-honest two-party setting, most of PPML protocols focus on ML inference except for SecureML [24] and QUOTIENT [18]. Nevertheless, SecureML and QUOTIENT only implemented the small dataset MNIST that has 60,000 training samples and 10 different classes. For two-party ML inference with semi-honest security, the state-of-the-art PPML protocol CrypTFlow2 [28] is able to perform private inference over complex deep neural networks (DNNs) like ResNet-50 (50 layers, 23.5 million parameters) and DenseNet-121 (121 layers, 8.5 million parameters), which can be trained over a large-scale dataset ImageNet that contains more than 1,000,000 training samples and 1000 different classes. Their implementation [28] needs about 546 *s* and 32 GB of communication for ResNet-50, and 463 *s* and 35 GB of communication for DenseNet-121. In the two-party setting, it seems to have been highly efficient for ML inference with semi-honest security, but the ML inference against malicious adversaries and the ML training still have a low efficiency, which needs to be addressed in the future work.

In the honest-majority setting, the known PPML protocols only consider the three-party and four-party cases tolerate one corruption. In this setting, we can achieve a relatively high efficiency for private inference and training. By accelerating semi-honest 3PC with GPU, CryptGPU [31] can perform one private inference over the ImageNet-scale ResNet-50 (resp., ResNet-152) using 9.3 *s* and 3.1 GB of communication (resp., 25.8 *s* and 6.6 GB of communication). The private training implemented by CryptGPU is able to support VGG-16 (16 layers, 138 million parameters), which is trained over a Tiny ImageNet dataset that contains 100,000 training samples and more than 200 different classes. Their implementation reports the running time and communication for a single iteration of private training, which are 13.9 *s* and 7.6 GB respectively. For malicious security, the best-known PPML protocol Falcon [292] can run a private inference over a neural-network VGG-16 trained with Tiny ImageNet using 12.1 *s* of running time and 0.4 GB of communication. However, Falcon with malicious security takes over 3 years and about 1012 TB of communication to train a VGG-16 model over the Tiny ImageNet dataset. When a majority of parties are honest, multiple PPML protocols can also achieve stronger security property than security with abort (*i.e.*, fairness and GOD) using a small overhead. In this case, the PPML protocols in the four-party setting have a better performance than those in the three-party setting, but require a stronger assumption about the number of honest parties. Among these PPML protocols achieving fairness or GOD, Tetrad [294] has the best efficiency for now. Particularly, Tetrad takes 183 *s* and 35 GB of communication to train a VGG-16 model over a small dataset CIFAR-10 that includes 50,000 training samples and 10 different classes. Overall, in the three-party/four-party setting, private inference has been practical and can scale to complex models and large datasets, even in the presence of malicious adversaries. In the same setting, private training provides a high efficiency and supports a moderate-sized dataset for semi-honest security, but has a very low efficiency for malicious security. Besides, it will be an interesting future work to design honest-majority PPML protocols with at least five parties and two corrupted parties.

For ML applications, we need to handle multiple different-type functions. For example, in DNNs, we need to compute Matrix Multiplication, Convolution *etc.*, for linear layers, and ReLU, Max Pooling, Sigmoid, SoftMax *etc.*, for non-linear layers. Therefore, we need to construct mixed-mode MPC protocols, which support both arithmetic circuits and Boolean circuits and allow to convert between arithmetic and Boolean circuits. Additionally, for division operations or a function represented as a circuit that has a large depth, we may need to use the garbled circuit approach to achieve better efficiency. In the dishonest-majority setting, the ABY-like protocols [7, 25] developed the techniques to realize the conversion among arithmetic sharing, Boolean sharing and Yao's GCs in the presence of semi-honest adversaries. The ABY-like protocols focus on the case that the circuit evaluation is executed between two parties. In the multi-party setting, the recent work [295] proposes the semi-honest protocols to support arithmetic sharing, Boolean sharing, Yao's GCs, and conversions between any two represents. However, their protocols need a trusted party to distribute all correlated randomness among the parties evaluating the circuits, which makes the protocols have a weaker security guarantee. For malicious security, the conversion between distributed garbled circuits and SPDZ-style authenticated sharings can be realized using the doubly authenticated bits (daBits) technique [158, 296–298] or the more efficient extended daBits (edaBits) technique [299]. Specifically, a daBit consists of a pair of random sharings $([r]_2, [r]_M)$, where $r \in \{0, 1\}$ and either $M = p$ for a prime p or $M = 2^k$. The daBits technique was first presented

by Rotaru and Wood [298] for the case of $M = p$. Then, the performance was further improved for the case of $M = p$ in [296, 297, 299], and daBits over a ring $\mathbb{Z}_{2^k}$ were also shown in [158, 299], where the known implementation [296] takes about 11.8 KB for generating one daBit with a large prime p and can generate about 2150 daBits per second in the two-party malicious setting. The state-of-the-art edaBits protocol [299] adopts the "Cut-and-Bucketing" technique to check the consistency of values in two different domains, where an edaBit consists of a set of random sharings $([r_{m-1}]_2, \ldots, [r_0]_2)$ in the binary domain and a sharing $[r]_M$ in the arithmetic domain, such that $r = \sum_{i\in[0,m)} r_i \cdot 2^i \mod M$. In the two-party malicious setting, the edaBits approach will reduce the communication cost by a factor of 2× for implementing comparison of 63-bit integers, compared to the daBits technique [299]. The "Cut-and-Bucketing" technique for malicious security leads to at least a factor of 3 overhead over the semi-honest protocol. It is an interesting open problem to construct a concretely efficient edaBits protocol with malicious security, which achieves an overhead of 2 or even smaller. In the honest-majority setting, the protocols against malicious adversaries, which allow to convert between the arithmetic, Boolean, and garbling worlds, can be constructed more efficiently [19, 294], where the techniques underlying the constant-round MPC protocols (e.g., [236, 239]) can be used and adapted.

On the other hand, the recent studies by Boyle *et al.* [300, 301] proposed a new approach to construct mixed-mode MPC protocols based on function secret sharing (FSS), which is useful for ML applications with optimal online communication and round complexity. Their FSS approach supports arithmetic operations that are mixed with non-arithmetic operations. In particular, for a non-arithmetic function g such as ReLU, two parties can obtain two succinct FSS keys to evaluate function $g_r(x) = g(x + r)$ where r is a randomness shared by the parties. In general, the FSS-based approach requires more communication in the preprocessing phase than the GC or GMW approach, unless the FSS keys are distributed by a trusted dealer, or the input length is relatively small. This naturally leaves a future work to reduce the preprocessing cost for distributing the FSS keys by a concretely efficient 2PC protocol. For online communication cost and rounds, the FSS-based approach outperforms the GC and GMW approaches. Their FSS-based approach can be also secure against malicious adversaries [300]. For now, Boyle *et al.* [300, 301] only proposed efficient constructions for functions including comparison (e.g., ReLU), splines (*e.g.*, used in sigmoid), bit-decomposition, zero test, and arithmetic/logical shifts. There are still many functions used in ML and scientific computation (e.g., exponentiation, tanh, and reciprocal of square root), whose concretely efficient FSS-based 2PC protocols are unknown. Besides, Boyle *et al.* [300, 301] only gave two-party constructions. It is an interesting future work to construct concretely efficient FSS-based MPC protocols with optimal online communication and rounds for multiple parties (*i.e.*, $n \geq 3$). While prior work uses a uniform bitwidth for the whole ML inference, the recent work by Rathee *et al.* [27] proposed the mixed bitwidths approach, *i.e.,* operating in low bitwidths and going to high bitwidths only when necessary. They designed new protocols to switch between bitwidths and operations on values of differing bitwidths. Their approach is interesting and able to obtain better efficiency. While the work [27] only considers private ML inference in the two-party setting, it is worth further developing the mixed bitwidths approach to private ML training and the multi-party setting.

7 Conclusion and future work

We have described the (recent) development of concretely efficient MPC protocols along with the key techniques underlying these MPC protocols. Particularly, we present the high-level ideas in the recent MPC protocols and OT/OLE protocols. As an example of MPC applications, we discuss privacy-preserving machine learning, and summarize related work as well as conversion and FSS-based techniques. It is desired that this survey will help new researchers (who are interesting for MPC) understand the recent development of concretely efficient MPC rapidly, and to preliminarily understand some key techniques as a starting point of MPC study.

To deploy MPC on a large scale, standardization is a necessary step. Nevertheless, this is *not* an easy task, as there exists many different kinds of MPC protocols that have different advantages in terms of security and efficiency. Besides, there are many techniques and different assumptions that are used in the design of MPC. These make the MPC standardization procedure becomes hard. Of course, we can first standardize a batch of MPC protocols in the same setting, and then standardize the next batch in the other setting. When standardization is a long-time procedure and needs to take a large amount of

financial resources, this approach is very expensive. Furthermore, how to keep compatibility of multiple MPC protocols in different standardization procedures is a problem. All of these need to be addressed and solved in the future work. Recently, ISO is preparing the standardization process for MPC based on secret sharing [302]. Besides, NIST will standardize multi-party threshold cryptography in the future [303], where MPC is a key technique to realize AES encryption/decryption, EdDSA signing, the distributed key-generation of RSA, *etc.* NIST also intends to accompany the progress of emerging technologies in the area of privacy enhancing cryptography [304], which includes MPC, ZK, HE, *etc.*

We have summarized some open problems and future work in the previous sections. In the following, we conclude this work by further listing several open problems and future work for concretely efficient MPC protocols.

- **Constant-round 2PC:** The recent break-through work by Rosulek and Roy [199] reduced the size of garbled circuits from 2κ bits per AND gate to $3\kappa/2+5$ bits per AND gate. A natural open problem is whether one can do better, e.g., about $4\kappa/3$ bits per AND gate while keeping compatibility with free-XOR and high computational efficiency. If this seems to be impossible, one can attempt to prove that $\approx 3\kappa/2$ bits per AND gate is optimal in a more inclusive model than the linear garbling model in [195]. When the work [199] focuses on the semi-honest adversary, another open problem is to extend the slicing-and-dicing technique in [199] to two-party distributed garbling that can be used to design maliciously secure 2PC protocols by combining with BDOZ-style IT-MACs.
- **Constant-round MPC:** We conclude three future work for designing constant-round MPC.
  (a) *Dishonest majority for garbling:* For multi-party distributed garbling based on only symmetric primitives, the state-of-the-art technique by Yang *et al.* [110] achieves $(4n-6)|\mathcal{C}|\kappa$ bits in terms of the size of a garbled circuit. It is a challenging task to further reduce it to about $2n|\mathcal{C}|\kappa$ bits based on still symmetric primitives. In other words, is it possible to totally apply the half-gate technique to multi-party distributed garbling?
  (b) *Dishonest majority for AND triples:* Currently, we use a TinyOT-like protocol to generate authenticated AND triples, which requires an overhead of at least 3 to achieve malicious security over the corresponding semi-honest protocol, where the overhead is from the usage of the bucketing technique. It is an interesting open problem that designs an authenticated-AND-triple protocol achieving an overhead of 2 (or even smaller) using a novel technique.
  (c) *Honest majority:* If the number of parties $n > 5$, Chandran *et al.* [237] presented a constant-round MPC protocol with corruption threshold $t \leq \sqrt{n}$. It is an interesting future work that constructing a constant-round concrete-efficient MPC protocol tolerating $t < n/2$ corrupted parties when $n > 5$.
- **SPDZ:** The efficiency bottleneck for SPDZ-style protocols is to generate authenticated triples over a large field. The state-of-the-art protocol [161] based on a variant of the ring-LPN assumption obtains a relatively low communication complexity, which is two orders of magnitude smaller than Overdrive [152]. However, this protocol has a computation complexity of $O(N \log N)$ for encoding (while fast Fourier transform (FFT) is used) which is large for large N, where N is the number of resulting authenticated triples. An important future work is to reduce the computation complexity while keeping small communication complexity for the ring-LPN-based protocol.
- **LPN variants for MPC:** COT and OLE as well as their variants are key building blocks for MPC in the dishonest-majority setting. To design the COT and (V)OLE protocols with low communication, several LPN variants have already been proposed, including LPN with a regular noise distribution [114, 132], LPN with static leakage [132], ring-LPN with reducible polynomials and a regular noise distribution [161]. An important future work is to further analyze the LPN variants proposed in the MPC context, which allows to establish more confidence on the hardness of these LPN problems.
- **Large-scale MPC with honest majority:** For honest-majority MPC in the malicious setting, several recent work [178, 179, 184] designed large-scale MPC protocols, which scales practically to hundreds of thousands of parties. However, their concrete efficiency is still not high. Constructing large-scale maliciously secure MPC protocols with higher concrete efficiency as well as giving an efficient implementation scaling to thousands of parties will be an interesting future work.

Conflict of Interest

The authors declare that they have no conflict of interest.

Data Availability

No data are associated with this article.

Authors' Contributions

Dengguo Feng mainly surveyed the MPC protocols based on linear secret sharing schemes and the privacy-preserving machine learning protocols building upon MPC techniques. Kang Yang mainly surveyed constant-round MPC protocols and the oblivious transfer and oblivious linear-function evaluation protocols. The authors discussed the recent development and future work of MPC, and then jointly wrote this paper.

Acknowledgements

We thank the anonymous reviewers for their helpful comments.

Funding

This work was supported in part by the National Key Research and Development Program of China (Grant No. 2018YFB0804105), and in part by the National Natural Science Foundation of China (Grant Nos. 62102037, 61932019).

References

[1] Beaver D, Micali S and Rogaway P. The round complexity of secure protocols (extended abstract). In: 22nd ACM STOC. ACM Press, 1990, 503–13.
[2] Ben-Or M, Goldwasser S and Wigderson A. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In: 20th ACM STOC. ACM Press, 1988, 1–10.
[3] Chaum D, Crépeau C and Damgård I. Multiparty unconditionally secure protocols (extended abstract). In: 20th ACM STOC. ACM Press, 1988, 11–19.
[4] Goldreich O, Micali S and Wigderson A. How to play any mental game or A completeness theorem for protocols with honest majority. In: Aho A (ed.). 19th ACM STOC. ACM Press, 1987, 218–29.
[5] Rabin T and Ben-Or M. Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In: 21st ACM STOC. ACM Press, 1989, 73–85.
[6] Yao AC-C. How to generate and exchange secrets (extended abstract). In: 27th FOCS. IEEE Computer Society Press, 1986, 162–7.
[7] Demmler D, Schneider T and Zohner M. ABY – A framework for efficient mixed-protocol secure two-party computation. In: NDSS 2015. The Internet Society, 2015.
[8] Wang X, Malozemoff AJ and Katz J. EMP-toolkit: Efficient MultiParty Computation Toolkit. https://github.com/emp-toolkit, 2016.
[9] Alexandra Institute. FRESCO – A FRamework for Efficient Secure COmputation. https://github.com/aicis/fresco.
[10] Multiparty.org Development Team. Javascript Implementation of Federated Functionalities, 2020. https://github.com/multiparty/jiff.
[11] Data61. Mp-spdz. https://github.com/data61/MP-SPDZ, 2019.
[12] Schoenmakers B. MPyC: Secure Multiparty Computation in Python https://github.com/lschoe/mpyc.
[13] Aly A, Keller M and Orsini E et al. SCALE-MAMBA v1.14: Documentation, 2021. https://github.com/KULeuven-COSIC/SCALE-MAMBA.
[14] Bogdanov D, Laur S and Willemson J. Sharemind: A framework for fast privacy-preserving computations. In: Jajodia S and López J (eds.). ESORICS 2008, volume 5283 of LNCS. Heidelberg: Springer, 2008, 192–206.
[15] Songhori EM, Hussain SU and Sadeghi A-R et al. TinyGarble: highly compressed and scalable sequential garbled circuits. In: 2015 IEEE Symposium on Security and Privacy. IEEE Computer Society Press, 2015, 411–28.
[16] Hastings M, Hemenway B and Noble D et al. SoK: General purpose compilers for secure multi-party computation. In: 2019 IEEE Symposium on Security and Privacy. IEEE Computer Society Press, 2019, 1220–37.
[17] Keller M. MP-SPDZ: A versatile framework for multi-party computation. In: Ligatti J, Ou X, Katz J and Vigna G (eds.). ACM CCS 20. ACM Press, 2020, 1575–90.
[18] Agrawal N, Shahin Shamsabadi A and Kusner MJ et al. QUOTIENT: Two-party secure neural network training and prediction. In: Cavallaro L, Kinder J, Wang X, Katz J (eds.). ACM CCS 2019. ACM Press, 2019, 1231–47.
[19] Chaudhari H, Rachuri R and Suresh A. Trident: Efficient 4PC framework for privacy preserving machine learning. In: NDSS 2020. The Internet Society, 2020.
[20] Juvekar C, Vaikuntanathan V and Chandrakasan A. GAZELLE: A low latency framework for secure neural network inference. In: Enck W and Felt AP (eds.). USENIX Security 2018. USENIX Association, 2018, 1651–69.
[21] Kumar N, Rathee M and Chandran N et al. CrypTFlow: secure TensorFlow inference. In: 2020 IEEE Symposium on Security and Privacy. IEEE Computer Society Press, 2020, 336–53.
[22] Mishra P, Lehmkuhl R and Srinivasan A et al. Delphi: A cryptographic inference service for neural networks. In: Capkun S and Roesner F (eds.). USENIX Security 2020. USENIX Association, 2020, 2505–22.
[23] Mohassel P and Rindal P. ABY^3: A mixed protocol framework for machine learning. In: Lie D, Mannan M, Backes M and Wang XF (eds.). ACM CCS 2018. ACM Press, 2018, 35–52.
[24] Mohassel P and Zhang Y. SecureML: A system for scalable privacy-preserving machine learning. In: 2017 IEEE Symposium on Security and Privacy. IEEE Computer Society Press, 2017, 19–38.

[25] Patra A, Schneider T and Suresh A et al. ABY2.0: Improved Mixed-protocol Secure Two-party Computation. Cryptology ePrint Archive, Report 2020/1225, 2020. https://eprint.iacr.org/2020/1225.
[26] Patra A and Suresh A. BLAZE: Blazing fast privacy-preserving machine learning. In: NDSS 2020. The Internet Society, 2020.
[27] Rathee D, Rathee M and Goli RKK et al. SIRNN: A Math Library for Secure RNN Inference. Cryptology ePrint Archive, Report 2021/459, 2021. https://eprint.iacr.org/2021/459.
[28] Rathee D, Rathee M and Kumar N et al. CrypTFlow2: practical 2-party secure inference. In: Ligatti J, Ou X, Katz J and Vigna G (eds.). ACM CCS 20. ACM Press, 2020, 325–42.
[29] Riazi M S, Samragh M and Chen H et al. XONN: XNOR-based oblivious deep neural network inference. In: Heninger N and Traynor P (eds.). USENIX Security 2019. USENIX Association, 2019, 1501–18.
[30] Schoppmann P, Gascón A and Raykova M et al. Make some ROOM for the zeros: data sparsity in secure distributed machine learning. In: Cavallaro L, Kinder J, Wang XF and Katz J (eds.). ACM CCS 2019. ACM Press, 2019, 1335–50.
[31] Tan S, Knott B and Tian Y et al. CryptGPU: fast privacy-preserving machine learning on the GPU. In: IEEE Symposium on Security and Privacy, 2021.
[32] Brunetta C, Tsaloli G and Liang B et al. Non-interactive, secure verifiable aggregation for decentralized, privacy-preserving learning. Cryptology ePrint Archive, Report 2021/654, 2021. https://eprint.iacr.org/2021/654.
[33] Fereidooni H, Marchal S and Miettinen M et al. SAFELearn: Secure Aggregation for private FEderated Learning. Cryptology ePrint Archive, Report 2021/386, 2021. https://eprint.iacr.org/2021/386.
[34] Han K, Jeong J and Sohn JH et al. Efficient privacy preserving logistic regression inference and training. Cryptology ePrint Archive, Report 2020/1396, 2020. https://eprint.iacr.org/2020/1396.
[35] Zheng W, Deng R and Chen W et al. Cerebro: A Platform for Multi-party Cryptographic Collaborative Learning. Cryptology ePrint Archive, Report 2021/759, 2021. https://eprint.iacr.org/2021/759.
[36] Zheng Q, Popa RA and Gonzalez JE et al. Helen: maliciously secure coopetitive learning for linear models. In: 2019 IEEE Symposium on Security and Privacy. IEEE Computer Society Press, 2019, 724–38.
[37] Bogdanov D, Niitsoo M and Toft T et al. High-performance secure multi-party computation for data mining applications. Int J Inf Secur 2012; **11**: 403–18.
[38] Burkhart M, Strasser M and Many D et al. SEPIA: Privacy-preserving aggregation of multi-domain network events and statistics. In: USENIX Security 2010. USENIX Association, 2010, 223–40.
[39] Cramer R, Damgård IB and Nielsen JB. Secure Multiparty Computation and Secret Sharing. Cambridge University Press, 2015.
[40] Lindell Y and Pinkas B. Privacy preserving data mining. J Cryptol 2002; **15**: 177–206.
[41] Ben-David A, Nisan N and Pinkas B. FairplayMP: A system for secure multi-party computation. In: Ning P, Syverson PF and Jha S (eds.). ACM CCS 2008. ACM Press, 2008, 257–66.
[42] Bogetoft P, Christensen DL and Damgård I et al. Secure multiparty computation goes live. In: Dingledine R and Golle P (eds.). FC 2009, volume 5628 of LNCS. Heidelberg: Springer, 2009, 325–43
[43] Keller M, Orsini E and Scholl P. MASCOT: Faster malicious arithmetic secure computation with oblivious transfer. In: Weippl ER, Katzenbeisser S, Kruegel C, Myers AC and Halevi S (eds.). ACM CCS 2016. ACM Press, 2016, 830–42.
[44] Cho H, Wu DJ and Berger B. Secure genome-wide association analysis using multiparty computation. Nat Biotechnol 2018; **36**: 547–51.
[45] Jagadeesh KA, Wu DJ and Birgmeier JA et al. Deriving genomic diagnoses without revealing patient genomes. Science 2017; **357**: 692–5.
[46] Jha S, Kruger L and Shmatikov V. Towards practical privacy for genomic computation. In: 2008 IEEE Symposium on Security and Privacy. IEEE Computer Society Press, 2008, 216–30.
[47] Archer DW, Bogdanov D and Lindell Y et al. From keys to databases – real-world applications of secure multi-party computation. Comput J 2018; **61**: 1749–71.
[48] Almashaqbeh G and Solomon R. Sok: Privacy-preserving computing in the blockchain era. Cryptology ePrint Archive, Report 2021/727, 2021. https://eprint.iacr.org/2021/727.
[49] Atapoor S, Smart NP and Alaoui YT. Private liquidity matching using MPC. Cryptology ePrint Archive, Report 2021/475, 2021. https://eprint.iacr.org/2021/475.
[50] Banerjee A, Clear M and Tewari H. zkhawk: Practical private smart contracts from MPC-based hawk. Cryptology ePrint Archive, Report 2021/501, 2021. https://eprint.iacr.org/2021/501.
[51] Dolev S and Wang Z. Sodsmpc: FSM based anonymous and private quantum-safe smart contracts. Cryptology ePrint Archive, Report 2020/1346, 2020. https://eprint.iacr.org/2020/1346.
[52] El Defrawy K and Lampkins J. Founding digital currency on secure computation. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, CCS'14. Association for Computing Machinery, 2014, 1–14.
[53] Green M and Miers I. Bolt: Anonymous payment channels for decentralized currencies. In: Thuraisingham BM, Evans D, Malkin T and Xu D (eds.). ACM CCS 2017. ACM Press, 2017, 473–89.
[54] Ames S, Hazay C and Ishai Y et al. Ligero: Lightweight sublinear arguments without a trusted setup. In: Thuraisingham BM, Evans D, Malkin T, Xu D (eds.). ACM CCS 2017. ACM Press, 2017, 2087–2104.
[55] Bhadauria R, Fang Z and Hazay C et al. Ligero++: A new optimized sublinear IOP. In: Ligatti J, Ou X, Katz J and Vigna G (eds.). ACM CCS 20. ACM Press, 2020, 2025–38.
[56] Chase M, Derler D and Goldfeder S et al. Post-quantum zero-knowledge and signatures from symmetric-key primitives. In: Thuraisingham BM, Evans D, Malkin T and Xu D (eds.). ACM CCS 2017. ACM Press, 2017, 1825–1842
[57] De Saint Guilhem CD, Orsini E and Tanguy T. Limbo: Efficient zero-knowledge mpcith-based arguments. Cryptology ePrint Archive, Report 2021/215, 2021. https://ia.cr/2021/215.

[58] Giacomelli I, Madsen J and Orlandi C. ZKBoo: Faster zero-knowledge for Boolean circuits. In: Holz T and Savage S (eds.). USENIX Security 2016. USENIX Association, 2016, 1069–83.

[59] Gvili Y, Scheffler S and Varia M. Booligero: Improved sublinear zero knowledge proofs for Boolean circuits. Cryptology ePrint Archive, Report 2021/121, 2021. https://eprint.iacr.org/2021/121.

[60] Ishai Y, Kushilevitz E and Ostrovsky R et al. Zero-knowledge from secure multiparty computation. In: Johnson DS and Feige U (eds.). 39th ACM STOC. ACM Press, 2007, 21–30.

[61] Katz J, Kolesnikov V and Wang X. Improved non-interactive zero knowledge with applications to post-quantum signatures. In: Lie D, Mannan M, Backes M and Wang XF (eds.). ACM CCS 2018. ACM Press, 2018, 525–37.

[62] Baum C, Braun L and Munch-Hansen A et al. Appenzeller to brie: Efficient zero-knowledge proofs for mixed-mode arithmetic and $\mathbb{Z}_{2^k}$. Cryptology ePrint Archive, Report 2021/750, 2021. https://eprint.iacr.org/2021/750.

[63] Baum C, Malozemoff AJ and Rosen M et al. Mac'n'cheese: Zero-knowledge proofs for arithmetic circuits with nested disjunctions. Cryptology ePrint Archive, Report 2020/1410, 2020. https://eprint.iacr.org/2020/1410.

[64] Dittmer S, Ishai Y and Ostrovsky R. Line-point zero knowledge and its applications. Cryptology ePrint Archive, Report 2020/1446, 2020. https://eprint.iacr.org/2020/1446.

[65] Frederiksen TK, Nielsen JB and Orlandi C. Privacy-free garbled circuits with applications to efficient zero-knowledge. In: Oswald E and Fischlin M (eds.) EUROCRYPT 2015, Part II, volume 9057 of LNCS. Heidelberg: Springer, 2015, 191–219.

[66] Heath D and Kolesnikov V. Stacked garbling for disjunctive zero-knowledge proofs. In: Canteaut A and Ishai Y (eds.). EUROCRYPT 2020, Part III, volume 12107 of LNCS. Heidelberg: Springer, 2020, 569–98.

[67] Jawurek M, Kerschbaum F and Orlandi C. Zero-knowledge using garbled circuits: how to prove non-algebraic statements efficiently. In: Sadeghi A-R, Gligor VD and Yung M (eds.). ACM CCS 2013. ACM Press, 2013, 955–66.

[68] Kondi Y and Patra A. Privacy-free garbled circuits for formulas: size zero and information-theoretic. In: Katz J and Shacham H (eds.). CRYPTO 2017, Part I, volume 10401 of LNCS. Heidelberg: Springer, 2017, 188–222.

[69] Weng C, Yang K, Katz J and Wang X. Wolverine: Fast, Scalable, and Communication-efficient Zero-knowledge Proofs for Boolean and Arithmetic Circuits. IEEE Computer Society Press, 2021.

[70] Weng C, Yang K and Xie X et al. Mystique: Efficient Conversions for Zero-knowledge Proofs with Applications to Machine Learning. Cryptology ePrint Archive, Report 2021/730, 2021. https://eprint.iacr.org/2021/730.

[71] Yang K, Sarkar P and Weng C et al. Quicksilver: Efficient and Affordable Zero-knowledge Proofs for Circuits and Polynomials Over Any Field. Cryptology ePrint Archive, Report 2021/076, 2021. https://eprint.iacr.org/2021/076.

[72] Canetti R, Gennaro R and Goldfeder S et al. UC non-interactive, proactive, threshold ECDSA with identifiable aborts. In: Ligatti J, Ou X, Katz J and Vigna G (eds.). ACM CCS 20. ACM Press, 2020, 1769–87.

[73] Chen M, Cohen R and Doerner J et al. Multiparty generation of an RSA modulus. In: Micciancio D and Ristenpart T (eds.). CRYPTO 2020, Part III, volume 12172 of LNCS. Heidelberg: Springer, 2020, 64–93.

[74] Chen M, Hazay C and Ishai Y et al. Diogenes: Lightweight scalable RSA modulus generation with a dishonest majority. Cryptology ePrint Archive, Report 2020/374, 2020. https://eprint.iacr.org/2020/374.

[75] Doerner J, Kondi Y and Lee E et al. Secure two-party threshold ECDSA from ECDSA assumptions. In: 2018 IEEE Symposium on Security and Privacy. IEEE Computer Society Press, 2018, 980–997.

[76] Doerner J, Kondi Y and Lee E et al. Threshold ECDSA from ECDSA assumptions: The multiparty case. In: 2019 IEEE Symposium on Security and Privacy. IEEE Computer Society Press, 2019, 1051–66.

[77] Frederiksen TK, Lindell Y and Osheter V et al. Fast distributed RSA key generation for semi-honest and malicious adversaries. In: Shacham H and Boldyreva A (eds.). CRYPTO 2018, Part II, volume 10992 of LNCS. Heidelberg: Springer, 2018, 331–61.

[78] Hazay C, Mikkelsen G and Rabin T et al. Efficient RSA key generation and threshold paillier in the two-party setting. J Cryptol 2019; **32**: 265–323.

[79] Lindell Y and Nof A. Fast secure multiparty ECDSA with practical distributed key generation and applications to cryptocurrency custody. In: Lie D, Mannan M, Backes M and Wang XF (eds.). ACM CCS 2018. ACM Press, 2018, 1837–54.

[80] Garimella G, Pinkas B and Rosulek M et al. Oblivious key-value stores and amplification for private set intersection. In: Malkin T and Peikert C (eds.). Advances in Cryptology – CRYPTO 2021, volume 12826 of LNCS. Springer International Publishing, 2021, 395–425.

[81] Pinkas B, Rosulek M and Trieu N et al. SpOT-light: lightweight private set intersection from sparse OT extension. In: Boldyreva A and Micciancio D (eds.). CRYPTO 2019, Part III, volume 11694 of LNCS. Heidelberg: Springer, 2019, 401–31.

[82] Pinkas B, Rosulek M and Trieu N et al. PSI from PaXoS: fast, malicious private set intersection. In: Canteaut A and Ishai Y (eds.). EUROCRYPT 2020, Part II, volume 12106 of LNCS. Heidelberg: Springer, 2020, 739–67.

[83] Pinkas B, Schneider T and Tkachenko O et al. Efficient circuit-based PSI with linear communication. In: Ishai Y and Rijmen V (eds.). EUROCRYPT 2019, Part III, volume 11478 of LNCS. Heidelberg: Springer, 2019, 122–53.

[84] Pinkas B, Schneider T and Weinert C et al. Efficient circuit-based PSI via cuckoo hashing. In: Nielsen JB and Rijmen V (eds.). EUROCRYPT 2018, Part III, volume 10822 of LNCS. Heidelberg: Springer, 2018, 125–57.

[85] Rindal P and Schoppmann P. VOLE-PSI: fast OPRF and Circuit-PSI from Vector-OLE. In: Canteaut A and Standaert F-X (eds.). Advances in Cryptology – EUROCRYPT 2021, volume 12697 of LNCS. Springer International Publishing, 2021, 901–30.

[86] Cleve R. Limits on the security of coin flips when half the processors are faulty (extended abstract). In: 18th ACM STOC. ACM Press, 1986, 364–69.

[87] Araki T, Furukawa J and Lindell Y et al. High-throughput semi-honest secure three-party computation with an honest majority. In: Weippl ER, Katzenbeisser S, Kruegel C, Myers AC, Halevi S (eds.). ACM CCS 2016. ACM Press, 2016, 805–17.

[88] Lindell Y. Secure multiparty computation. Commun ACM 2020; **64**: 86–96.

[89] Orsini E. Efficient, actively secure MPC with a dishonest majority: a survey. In: Bajard JC and Topuzoğlu A (eds.). International Workshop on the Arithmetic of Finite Fields – WAIFI 2020, volume 12542 of LNCS. Springer International Publishing, 2021, 42–71.
[90] Canetti R. Universally composable security: A new paradigm for cryptographic protocols. In: 42nd FOCS. IEEE Computer Society Press, 2001, 136–145.
[91] Goldreich O. Foundations of Cryptography: Volume 2 – Basic Applications. Cambridge University Press, 2004.
[92] Canetti C. Security and composition of multiparty cryptographic protocols. J Cryptol 2000; **13**: 143–202.
[93] Kushilevitz E, Lindell Y and Rabin T. Information-theoretically secure protocols and security under composition. In: Kleinberg JM (ed.). 38th ACM STOC. ACM Press, May 2006, 109–18.
[94] Goldwasser S and Lindell Y. Secure multi-party computation without agreement. J Cryptol 2005; **18**: 247–87.
[95] Even S, Goldreich O and Lempel A. A randomized protocol for signing contracts. Commun ACM 1985; **28**: 637–47.
[96] Rabin MO. How to Exchange Secrets by Oblivious Transfer. Technical Report TR-81, Aiken Computation Laboratory, Harvard University, 1981.
[97] Naor M and Pinkas B. Oblivious transfer and polynomial evaluation. In: 31st ACM STOC. ACM Press, 1999, 245–54.
[98] Shamir A. How to share a secret. Commun ACM 1979; 22: 612–3.
[99] Cramer R, Damgård I and Ishai Y. Share conversion, pseudorandom secret-sharing and applications to secure computation. In: Kilian J (ed.). TCC 2005, volume 3378 of LNCS. Heidelberg: Springer, 2005, 342–62.
[100] Ito M, Saito A and Nishizeki T. Secret sharing scheme realizing general access structure. Electron Commun Jpn III 1989; **72**: 56–64.
[101] Lindell Y and Nof A. A framework for constructing fast MPC over arithmetic circuits with malicious adversaries and an honest-majority. In: Thuraisingham BM, Evans D, Malkin T and Xu D (eds.). ACM CCS 2017. ACM Press, 2017, 259–76.
[102] Dessouky G, Koushanfar F and Sadeghi A-R et al. Pushing the communication barrier in secure computation using lookup tables. In: NDSS 2017. The Internet Society, 2017.
[103] Damgård I, Pastro V, Smart NP and Zakarias S. Multiparty computation from somewhat homomorphic encryption. In: Safavi-Naini R and Canetti R (eds.). CRYPTO 2012, volume 7417 of LNCS. Heidelberg: Springer, 2012, 643–62.
[104] Nielsen JB, Nordholt PS and Orlandi C et al. A new approach to practical active-secure two-party computation. In: Safavi-Naini R and Canetti R (eds.). CRYPTO 2012, volume 7417 of LNCS. Heidelberg: Springer, 2012, 681–700.
[105] Bendlin R, Damgård I and Orlandi C et al. Semi-homomorphic encryption and multiparty computation. In: Paterson KG (ed.). EUROCRYPT 2011, volume 6632 of LNCS. Heidelberg: Springer, 2011, 169–88
[106] Hazay C, Scholl P and Soria-Vazquez E. Low cost constant round MPC combining BMR and oblivious transfer. In: Takagi T and Peyrin T (eds.). ASIACRYPT 2017, Part I, volume 10624 of LNCS. Heidelberg: Springer, 2017, 598–628.
[107] Katz J, Ranellucci S and Rosulek M et al. Optimizing authenticated garbling for faster secure two-party computation. In: Shacham H and Boldyreva A (eds.). CRYPTO 2018, Part III, volume 10993 of LNCS. Heidelberg: Springer, 2018, 365–91.
[108] Wang X, Ranellucci S and Katz J. Authenticated garbling and efficient maliciously secure two-party computation. In: Thuraisingham BM, Evans D, Malkin T and Xu D (eds.). ACM CCS 2017. ACM Press, 2017, 21–37.
[109] Wang X, Ranellucci S and Katz J. Global-scale secure multiparty computation. In: Thuraisingham BM, Evans D, Malkin T and Xu D (eds.). ACM CCS 2017. ACM Press, 2017, 39–56.
[110] Yang K, Wang X and Zhang J. More efficient MPC from improved triple generation and authenticated garbling. In: Ligatti J, Ou X, Katz J and Vigna G (eds.). ACM CCS 20. ACM Press, 2020, 1627–46.
[111] Zhu R, Cassel D and Sabry A et al. NANOPI: extreme-scale actively-secure multi-party computation. In: Lie D, Mannan M, Backes M and Wang XF (eds.). ACM CCS 2018. ACM Press, 2018, 862–79.
[112] Damgård I, Nielsen JB, Nielsen M and Ranellucci S. The TinyTable protocol for 2-party secure computation, or: Gate-scrambling revisited. In: Katz J and Shacham H (eds.). CRYPTO 2017, Part I, volume 10401 of LNCS. Heidelberg: Springer, 2017, 167–87.
[113] Asharov G, Lindell Y and Schneider T et al. More efficient oblivious transfer and extensions for faster secure computation. In: Sadeghi A-R, Gligor VD, Yung M (eds.). ACM CCS 2013. ACM Press, 2013, 535–48.
[114] Hazay C, Orsini E and Scholl P et al. TinyKeys: A new approach to efficient multi-party computation. In: Shacham H and Boldyreva A (eds.). CRYPTO 2018, Part III, volume 10993 of LNCS. Heidelberg: Springer, 2018, 3–33.
[115] Schneider T and Zohner M. GMW vs. Yao? Efficient secure two-party computation with low depth circuits. In: Sadeghi A-R (ed.). FC 2013, volume 7859 of LNCS. Heidelberg: Springer, 2013, 275–92.
[116] Damgård I and Nielsen JB. Scalable and unconditionally secure multiparty computation. In: Menezes A (ed.). CRYPTO 2007, volume 4622 of LNCS. Heidelberg: Springer, 2007, 572–90.
[117] Gennaro R, Rabin MO and Rabin T. Simplified VSS and fast-track multiparty computations with applications to threshold cryptography. In: Coan BA and Afek Y (eds.). 17th ACM PODC. ACM, 1998, 101–111.
[118] Goyal V, Li H and Ostrovsky R et al. ATLAS: Efficient and scalable MPC in the honest majority setting. In: Advances in Cryptology – CRYPTO 2021. Springer, 2021.
[119] Goyal V and Song Y. Malicious security comes free in honest-majority MPC. Cryptology ePrint Archive, Report 2020/134, 2020. https://eprint.iacr.org/2020/134.
[120] Genkin D, Ishai Y and Prabhakaran M et al. Circuits resilient to additive attacks with applications to secure computation. In: Shmoys DB (ed.). 46th ACM STOC. ACM Press, 2014, 495–504.
[121] Beaver D. Efficient multiparty protocols using circuit randomization. In: Feigenbaum J (ed.). CRYPTO'91, volume 576 of LNCS. Heidelberg: Springer, 1992, 420–32.
[122] Beerliová-Trubíniová Z and Hirt M. Perfectly-secure MPC with linear communication complexity. In: Canetti R (ed.). TCC 2008, volume 4948 of LNCS. Heidelberg: Springer, 2008, 213–30

[123] Lindell Y, Oxman E and Pinkas B. The IPS compiler: optimizations, variants and concrete efficiency. In: Rogaway P (ed.). CRYPTO 2011, volume 6841 of LNCS. Heidelberg: Springer, 2011, 259–76.
[124] Boneh D, Boyle E and Corrigan-Gibbs H et al. Zero-knowledge proofs on secret-shared data via fully linear PCPs. In: Boldyreva A and Micciancio D (eds.). CRYPTO 2019, Part III, volume 11694 of LNCS. Heidelberg: Springer, 2019, 67–97.
[125] Boyle E, Gilboa N and Ishai Y et al. Efficient fully secure computation via distributed zero-knowledge proofs. In: Advances in Cryptology – ASIACRYPT 2020, volume 12493 of LNCS. Springer International Publishing, 2020, 244–76.
[126] Dalskov A, Escudero D and Keller M. Fantastic four: Honest-majority four-party secure computation with malicious security. Cryptology ePrint Archive, Report 2020/1330, 2020. https://eprint.iacr.org/2020/1330.
[127] Abspoel M, Cramer R and Damgård I et al. Efficient information-theoretic secure multiparty computation over $\mathbb{Z}/p^k\mathbb{Z}$ via galois rings. In: Hofheinz D and Rosen R (eds.). TCC 2019, Part I, volume 11891 of LNCS. Heidelberg: Springer, 2019, 471–501.
[128] Mouchet C, Troncoso-Pastoriza J and Bossuat J-P et al. Multiparty Homomorphic Encryption from Ring-learning-with-errors. Cryptology ePrint Archive, Report 2020/304, 2020. https://ia.cr/2020/304.
[129] Ben-Efraim A, Nielsen M and Omri E. Turbospeedz: Double your online SPDZ! Improving SPDZ using function dependent preprocessing. In: Deng RH, Gauthier-Umaña V, Ochoa M and Yung M (eds.). ACNS 19, volume 11464 of LNCS. Heidelberg: Springer, 2019, 530–49.
[130] Ishai Y, Kilian J and Nissim K et al. Extending oblivious transfers efficiently. In: Boneh D (ed.). CRYPTO 2003, volume 2729 of LNCS. Heidelberg: Springer, August 2003, 145–61.
[131] Hazay C, Orsini E and Scholl P et al. Concretely efficient large-scale MPC with active security (or, TinyKeys for TinyOT). In: Peyrin T and Galbraith S (eds.). ASIACRYPT 2018, Part III, volume 11274 of LNCS. Heidelberg: Springer, 2018, 86–117.
[132] Boyle E, Couteau G and Gilboa N et al. Efficient two-round OT extension and silent non-interactive secure computation. In: Cavallaro L, Kinder J, Wang X F and Katz J (eds.). ACM CCS 2019. ACM Press, 2019, 291–308.
[133] Rindal P, Raghuraman S and Couteau G. Silver: silent VOLE and oblivious transfer from hardness of decoding structured LDPC codes. In: Advances in Cryptology – CRYPTO 2021, volume 12827 of LNCS. Springer International Publishing, 2021, 502–34.
[134] Yang K, Weng C, Lan X and et al. Ferret: fast extension for correlated OT with small communication. In: Ligatti J, Ou X, Katz J and Vigna G (eds.). ACM CCS 20. ACM Press, 2020, 1607–26.
[135] Asharov G, Lindell Y and Schneider T et al. More efficient oblivious transfer extensions with security for malicious adversaries. In: Oswald E and Fischlin M (eds.). EUROCRYPT 2015, Part I, volume 9056 of LNCS. Heidelberg: Springer, 2015, 673–701.
[136] Keller M, Orsini E and Scholl P. Actively secure OT extension with optimal overhead. In: Gennaro R and Robshaw MJB (eds.). CRYPTO 2015, Part I, volume 9215 of LNCS. Heidelberg: Springer, 2015, 724–41.
[137] Goyal V, Song Y and Zhu C. Guaranteed output delivery comes free in honest majority MPC. In: Micciancio D and Ristenpart T (eds.). CRYPTO 2020, Part II, volume 12171 of LNCS. Heidelberg: Springer, 2020, 618–46.
[138] Chida K, Genkin D and Hamada K et al. Fast large-scale honest-majority MPC for malicious adversaries. In: Shacham H and Boldyreva A (eds.). CRYPTO 2018, Part III, volume 10993 of LNCS. Heidelberg: Springer, 2018, 34–64.
[139] Furukawa J and Lindell Y. Two-thirds honest-majority MPC for malicious adversaries at almost the cost of semi-honest. In: Cavallaro L, Kinder J, Wang XF and Katz J (eds.). ACM CCS 2019. ACM Press, 2019, 1557–71.
[140] Nordholt PS and Veeningen M. Minimising communication in honest-majority MPC by batchwise multiplication verification. In: Preneel B and Vercauteren F (eds.). ACNS 18, volume 10892 of LNCS. Heidelberg: Springer, 2018, 321–39.
[141] Abspoel M, Cramer R and Escudero D et al. Improved single-round secure multiplication using regenerating codes. Cryptology ePrint Archive, Report 2021/253, 2021. https://eprint.iacr.org/2021/253.
[142] Guruswami V and Wootters M. Repairing Reed-Solomon codes. In: Wichs D and Mansour Y (eds.). 48th ACM STOC. ACM Press, 2016, 216–26.
[143] Keller M, Rotaru D and Smart NP et al. Reducing communication channels in MPC. In: Catalano D and De Prisco R (eds.). SCN 18, volume 11035 of LNCS. Heidelberg: Springer, 2018, 181–99.
[144] Smart NP and Wood T. Error detection in monotone span programs with application to communication-efficient multi-party computation. In: Matsui M (ed.). CT-RSA 2019, volume 11405 of LNCS. Heidelberg: Springer, 2019, 210–29.
[145] Ishai Y, Prabhakaran M and Sahai A. Founding cryptography on oblivious transfer – efficiently. In: Wagner D (ed.). CRYPTO 2008, volume 5157 of LNCS. Heidelberg: Springer, 2008, 572–91.
[146] Hazay C, Ishai Y and Marcedone A et al. LevioSA: Lightweight secure arithmetic computation. In: Cavallaro L, Kinder J, Wang XF and Katz J (eds.). ACM CCS 2019. ACM Press, 2019, 327–44.
[147] Hazay C, Venkitasubramaniam M and Weiss M. The price of active security in cryptographic protocols. In: Canteaut A and Ishai Y (eds.). EUROCRYPT 2020, Part II, volume 12106 of LNCS. Heidelberg: Springer, 2020, 184–215.
[148] Chen H and Cramer R. Algebraic geometric secret sharing schemes and secure multi-party computations over small fields. In: Dwork C (ed.). CRYPTO 2006, volume 4117 of LNCS. Heidelberg: Springer, 2006, 521–536.
[149] Damgård I, Keller M and Larraia E et al. Practical covertly secure MPC for dishonest majority – or: Breaking the SPDZ limits. In: Crampton J, Jajodia S and Mayes K (eds.). ESORICS 2013, volume 8134 of LNCS. Heidelberg: Springer, 2013, 1–18.
[150] Gilboa N. Two party RSA key generation. In: Wiener MJ (ed.). CRYPTO'99, volume 1666 of LNCS. Heidelberg: Springer, 1999, 116–29.
[151] Brakerski Z, Gentry C and Vaikuntanathan V. (Leveled) fully homomorphic encryption without bootstrapping. In: Goldwasser S (ed.). ITCS 2012. ACM, 2012, 309–325.

[152] Keller M, Pastro V and Rotaru D. Overdrive: making SPDZ great again. In: Nielsen JB and Rijmen V (eds.). EUROCRYPT 2018, Part III, volume 10822 of LNCS. Heidelberg: Springer, 2018, 158–89.
[153] Baum C, Cozzo D and Smart NP. Using TopGear in overdrive: A more efficient ZKPoK for SPDZ. In: Paterson KG and Stebila D (eds.). SAC 2019, volume 11959 of LNCS. Heidelberg: Springer, 2019, 274–302.
[154] Chen H, Kim M and Razenshteyn I et al. Maliciously secure matrix multiplication with applications to private deep learning. Cryptology ePrint Archive, Report 2020/451, 2020. https://eprint.iacr.org/2020/451.
[155] Brakerski Z. Fully homomorphic encryption without modulus switching from classical GapSVP. In: Safavi-Naini R and Canetti R (eds.). CRYPTO 2012, volume 7417 of LNCS. Heidelberg: Springer, 2012, 868–86.
[156] Fan J and Vercauteren F. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144, 2012. http://eprint.iacr.org/2012/144.
[157] Cramer R, Damgård I and Escudero D et al. SPD $\mathbb{Z}_{2^k}$: Efficient MPC mod 2^k for dishonest majority. In: Shacham H and Boldyreva A (eds.). CRYPTO 2018, Part II, volume 10992 of LNCS. Heidelberg: Springer, 2018, 769–98.
[158] Damgård I, Escudero D and Frederiksen TK et al. New primitives for actively-secure MPC over rings with applications to private machine learning. In: 2019 IEEE Symposium on Security and Privacy. IEEE Computer Society Press, 2019, 1102–20.
[159] Catalano D, Di Raimondo M and Fiore D et al. Mon$\mathbb{Z}_{2^k}$a: Fast maliciously secure two party computation on $\mathbb{Z}_{2^k}$. In: Kiayias A, Kohlweiss M, Wallden P and Zikas V (eds.). PKC 2020, Part II, volume 12111 of LNCS. Heidelberg: Springer, 2020, 357–86.
[160] Orsini E, Smart NP and Vercauteren F. Overdrive2k: efficient secure MPC over $\mathbb{Z}_{2^k}$ from somewhat homomorphic encryption. In: Jarecki S (ed.). CT-RSA 2020, volume 12006 of LNCS. Heidelberg: Springer, 2020, 254–83.
[161] Boyle E, Couteau G and Gilboa N et al. Efficient pseudorandom correlation generators from ring-LPN. In: Micciancio D and Ristenpart T (eds.). CRYPTO 2020, Part II, volume 12171 of LNCS. Heidelberg: Springer, 2020, 387–416.
[162] Boyle E, Couteau G and Gilboa N et al. Efficient pseudorandom correlation generators: Silent OT extension and more. In: Boldyreva A and Micciancio D (eds.). CRYPTO 2019, Part III, volume 11694 of LNCS. Heidelberg: Springer, 2019, 489–518.
[163] Boyle E, Gilboa N and Ishai Y. Function secret sharing. In: Oswald E and Fischlin M (eds.). EUROCRYPT 2015, Part II, volume 9057 of LNCS. Heidelberg: Springer, 2015, 337–367.
[164] Frederiksen TK, Keller M and Orsini E et al. A unified approach to MPC with preprocessing using OT. In: Iwata T and Cheon JH (eds.). ASIACRYPT 2015, Part I, volume 9452 of LNCS. Heidelberg: Springer, 2015, 711–35.
[165] Larraia E, Orsini E and Smart N P. Dishonest majority multi-party computation for binary circuits. In: Garay JA and Gennaro R (eds.). CRYPTO 2014, Part II, volume 8617 of LNCS. Heidelberg: Springer, 2014, 495–512.
[166] Cascudo I, Gundersen J-S. A secret-sharing based MPC protocol for Boolean circuits with good amortized complexity. In: Theory of Cryptography, volume 12551 of LNCS. Springer International Publishing, 2020, 652–82.
[167] Damgård I, Lauritsen R and Toft T. An empirical study and some improvements of the MiniMac protocol for secure computation. In: Abdalla M and De Prisco R (eds.). SCN 14, volume 8642 of LNCS. Heidelberg: Springer, 2014, 398–415.
[168] Damgård I and Zakarias S. Constant-overhead secure computation of Boolean circuits using preprocessing. In: Sahai A (ed.). TCC 2013, volume 7785 of LNCS. Heidelberg: Springer, 2013, 621–41.
[169] Frederiksen TK, Pinkas B and Yanai A. Committed MPC – maliciously secure multiparty computation from homomorphic commitments. In: Abdalla M and Dahab R (eds.). PKC 2018, Part I, volume 10769 of LNCS. Heidelberg: Springer, 2018, 587–619.
[170] Cascudo I, Cramer R and Xing C et al. Amortized complexity of information-theoretically secure MPC revisited. In: Shacham H and Boldyreva A (eds.). CRYPTO 2018, Part III, volume 10993 of LNCS. Heidelberg: Springer, 2018, 395–426.
[171] Couteau G. A note on the communication complexity of multiparty computation in the correlated randomness model. In: Ishai Y and Rijmen V (eds.). EUROCRYPT 2019, Part II, volume 11477 of LNCS. Heidelberg: Springer, 2019, 473–503.
[172] Keller M, Orsini E and Rotaru D et al. Faster secure multi-party computation of AES and DES using lookup tables. In: Gollmann D, Miyaji A and Kikuchi H (eds.). ACNS 17, volume 10355 of LNCS. Heidelberg: Springer, 2017, 229–49.
[173] Furukawa J, Lindell Y and Nof A et al. High-throughput secure three-party computation for malicious adversaries and an honest majority. In: Coron JS and Nielsen JB (eds.). EUROCRYPT 2017, Part II, volume 10211 of LNCS. Heidelberg: Springer, 2017, 225–55.
[174] Araki T, Barak A and Furukawa J et al. Optimized honest-majority MPC for malicious adversaries – Breaking the 1 billion-gate per second barrier. In: 2017 IEEE Symposium on Security and Privacy. IEEE Computer Society Press, 2017, 843–62.
[175] Boyle E, Gilboa N and Ishai Y et al. Practical fully secure three-party computation via sublinear distributed zero-knowledge proofs. In: Cavallaro L, Kinder J, Wang XF and Katz J (eds.). ACM CCS 2019. ACM Press, 2019, 869–86.
[176] Ben-Sasson E, Fehr S and Ostrovsky R. Near-linear unconditionally-secure multiparty computation with a dishonest minority. In: Safavi-Naini R and Canetti R (eds.). CRYPTO 2012, volume 7417 of LNCS. Heidelberg: Springer, 2012, 663–80.
[177] Polychroniadou A and Song Y. Constant-overhead unconditionally secure multiparty computation over binary fields. In: Canteaut A and Standaert F-X (eds.). Advances in Cryptology – EUROCRYPT 2021, volume 12697 of LNCS. Springer International Publishing, 2021, 812–41.
[178] Beck G, Goel A and Jain A et al. Order-C secure multiparty computation for highly repetitive circuits. In: Advances in Cryptology – EUROCRYPT 2021, volume 12697 of LNCS. Springer International Publishing, 2021, 663–93.

[179] Gordon SD, Starin D and Yerukhimovich A. The more the merrier: Reducing the cost of large scale MPC. In: Advances in Cryptology – EUROCRYPT 2021, volume 12697 of LNCS. Springer International Publishing, 2021, 694–723.
[180] Franklin MK and Yung M. Communication complexity of secure computation (extended abstract). In: 24th ACM STOC. ACM Press, 1992, 699–710.
[181] Damgård I, Ishai Y and Krøigaard M. Perfectly secure multiparty computation and the computational overhead of cryptography. In: Gilbert H (ed.). EUROCRYPT 2010, volume 6110 of LNCS. Heidelberg: Springer, 2010, 445–65.
[182] Garay JA, Ishai Y and Ostrovsky R et al. The price of low communication in secure multi-party computation. In: Katz J and Shacham H (eds.). CRYPTO 2017, Part I, volume 10401 of LNCS. Heidelberg: Springer, 2017, 420–46.
[183] Genkin D, Ishai Y and Polychroniadou A. Efficient multi-party computation: From passive to active security via secure SIMD circuits. In: Gennaro R and Robshaw MJB (eds.). CRYPTO 2015, Part II, volume 9216 of LNCS. Heidelberg: Springer, 2015, 721–741.
[184] Goyal V, Polychroniadou A and Song Y. Unconditional communication-efficient MPC via Hall's marriage theorem. Cryptology ePrint Archive, Report 2021/834, 2021. https://eprint.iacr.org/2021/834.
[185] Escudero D and Dalskov A. Honest majority MPC with abort with minimal online communication. Cryptology ePrint Archive, Report 2020/1556, 2020. https://eprint.iacr.org/2020/1556.
[186] Ashur T, Cohen E and Hazay C et al. A new framework for garbled circuits. Cryptology ePrint Archive, Report 2021/739, 2021. https://eprint.iacr.org/2021/739.
[187] Bellare M, Hoang VT and Rogaway P. Foundations of garbled circuits. In: Yu T, Danezis G and Gligor VD (eds.). ACM CCS 2012. ACM Press, 2012, 784–96.
[188] Beaver D. Precomputing oblivious transfer. In: Coppersmith D (ed.). CRYPTO'95, volume 963 of LNCS. Heidelberg: Springer, 1995, 97–109.
[189] Huang Y, Evans D and Katz J et al. Faster secure two-party computation using garbled circuits. In: USENIX Security 2011. USENIX Association, 2011.
[190] Lindell Y and Pinkas B. A proof of security of Yao's protocol for two-party computation. J Cryptol 2009; **22**: 161–88.
[191] Malkhi D, Nisan N and Pinkas B et al. Fairplay – secure two-party computation system. In: Blaze M (ed.). USENIX Security 2004. USENIX Association, 2004, 287–302.
[192] Naor M, Pinkas B and Sumner R. Privacy preserving auctions and mechanism design. In: Proceedings of the 1st ACM Conference on Electronic Commerce – EC'99. New York, NY: ACM, 1999, 129–39.
[193] Pinkas B, Schneider T and Smart NP et al. Secure two-party computation is practical. In: Matsui M (ed.). ASIACRYPT 2009, volume 5912 of LNCS. Heidelberg: Springer, 2009, 250–67.
[194] Kolesnikov V and Schneider T. Improved garbled circuit: Free XOR gates and applications. In: Aceto L, Damgård I, Goldberg LA, Halldórsson MM, Ingólfsdóttir A and Walukiewicz I (eds.). ICALP 2008, Part II, volume 5126 of LNCS. Heidelberg: Springer, 2008, 486–498.
[195] Zahur S, Rosulek M and Evans D. Two halves make a whole – reducing data transfer in garbled circuits using half gates. In: Oswald E and Fischlin M (eds.). EUROCRYPT 2015, Part II, volume 9057 of LNCS. Heidelberg: Springer, 2015, 220–50.
[196] Choi SG, Katz J and Kumaresan R et al. On the security of the "free-XOR" technique. In: Cramer R (ed.). TCC 2012, volume 7194 of LNCS. Heidelberg: Springer, 2012, 39–53.
[197] Bellare M, Hoang VT and Keelveedhi S et al. Efficient garbling from a fixed-key blockcipher. In: IEEE Symposium on Security and Privacy. IEEE Computer Society Press, 2013, 478–92.
[198] Guo C, Katz J, Wang X and Yu Y. Efficient and secure multiparty computation from fixed-key block ciphers. In: 2020 IEEE Symposium on Security and Privacy. IEEE Computer Society Press, 2020, 825–41.
[199] Rosulek M and Roy L. Three halves make a whole? Beating the half-gates lower bound for garbled circuits. In: Malkin T and Peikert C (eds.). Advances in Cryptology – CRYPTO 2021, volume 12825 of LNCS. Springer International Publishing, 2021, 94–124.
[200] Kolesnikov V, Mohassel P and Rosulek M. FleXOR: Flexible garbling for XOR gates that beats free-XOR. In: Garay JA and Gennaro R (eds.). CRYPTO 2014, Part II, volume 8617 of LNCS. Heidelberg: Springer, 2014, 440–57.
[201] Gueron S, Lindell Y and Nof A et al. Fast garbling of circuits under standard assumptions. In: Ray I, Li N and Kruegel C (eds.). ACM CCS 2015. ACM Press, 2015, 567–78.
[202] Applebaum B, Ishai Y and Kushilevitz E. How to garble arithmetic circuits. In: Ostrovsky R (ed.). 52nd FOCS. IEEE Computer Society Press, 2011, 120–9.
[203] Ball M, Carmer B and Malkin T et al. Garbled neural networks are practical. Cryptology ePrint Archive, Report 2019/338, 2019. https://eprint.iacr.org/2019/338.
[204] Ball M, Malkin T and Rosulek M. Garbling gadgets for Boolean and arithmetic circuits. In: Weippl ER, Katzenbeisser S, Kruegel C, Myers AC and Halevi S (eds.). ACM CCS 2016. ACM Press, 2016, 565–77.
[205] Ben-Efraim A. On multiparty garbling of arithmetic circuits. In: Peyrin T and Galbraith S (eds.). ASIACRYPT 2018, Part III, volume 11274 of LNCS. Heidelberg: Springer, 2018, 3–33.
[206] Ben-Efraim A, Lindell Y and Omri E. Optimizing semi-honest secure multiparty computation for the Internet. In: Weippl ER, Katzenbeisser S, Kruegel C, Myers AC and Halevi S (eds.). ACM CCS 2016. ACM Press, 2016, 578–90.
[207] Aner Ben-Efraim A, Lindell Y and Omri E. Efficient scalable constant-round MPC via garbled circuits. In: Takagi T and Peyrin T (eds.). ASIACRYPT 2017, Part II, volume 10625 of LNCS. Heidelberg: Springer, 2017, 471–98.
[208] Ben-Efraim A, Cong K and Omri E et al. Large scale, actively secure computation from LPN and free-XOR garbled circuits. In: Advances in Cryptology – EUROCRYPT 2021, volume 12697 of LNCS. Springer International Publishing, 2021.
[209] Lindell Y and Pinkas B. An efficient protocol for secure two-party computation in the presence of malicious adversaries. In: Naor M (ed.). EUROCRYPT 2007, volume 4515 of LNCS. Heidelberg: Springer, 2007, 52–78.
[210] Afshar A, Mohassel P and Pinkas B et al. Non-interactive secure computation based on cut-and-choose. In: Nguyen PQ and Oswald E (eds.). EUROCRYPT 2014, volume 8441 of LNCS. Heidelberg: Springer, 2014, 387–404.

[211] Brandão LTAN. Secure two-party computation with reusable bit-commitments, via a cut-and-choose with forge-and-lose technique – (extended abstract). In: Sako K and Sarkar P (eds.). ASIACRYPT 2013, Part II, volume 8270 of LNCS. Heidelberg: Springer, 2013, 441–63.
[212] Frederiksen TK, Jakobsen TP and Nielsen JB. Faster maliciously secure two-party computation using the GPU. In: Abdalla M and De Prisco R (eds.). SCN 14, volume 8642 of LNCS. Heidelberg: Springer, 2014, 358–79.
[213] Huang Y, Katz J and Evans D. Efficient secure two-party computation using symmetric cut-and-choose. In: Canetti R and Garay JA (eds.). CRYPTO 2013, Part II, volume 8043 of LNCS. Heidelberg: Springer, 2013, 18–35.
[214] Huang Y, Katz Y and Kolesnikov V et al. Amortizing garbled circuits. In: Garay JA and Gennaro R (eds.). CRYPTO 2014, Part II, volume 8617 of LNCS. Heidelberg: Springer, 2014, 458–75.
[215] Kreuter B and Shen C-H. Billion-gate secure computation with malicious adversaries. In: Kohno T (ed.). USENIX Security 2012. USENIX Association, 2012, 285–300.
[216] Lindell Y. Fast cut-and-choose based protocols for malicious and covert adversaries. In: Canetti R and Garay J A (eds.). CRYPTO 2013, Part II, volume 8043 of LNCS. Heidelberg: Springer, 2013, 1–17.
[217] Lindell Y and Pinkas B. Secure two-party computation via cut-and-choose oblivious transfer. In: Yuval I (ed.). TCC 2011, volume 6597 of LNCS. Heidelberg: Springer, 2011, 329–46.
[218] Lindell Y and Riva B. Cut-and-choose Yao-based secure computation in the online/offline and batch settings. In: Garay JA and Gennaro R (eds.). CRYPTO 2014, Part II, volume 8617 of LNCS. Heidelberg: Springer, 2014, 476–494.
[219] Lindell Y and Riva B. Blazing fast 2PC in the offline/online setting with security for malicious adversaries. In: Ray I, Li N and Kruegel C (eds.). ACM CCS 2015. ACM Press, 2015, 579–590.
[220] Nielsen JB and Orlandi C. Cross and clean: amortized garbled circuits with constant overhead. In: Hirt M and Smith AD (eds.). TCC 2016-B, Part I, volume 9985 of LNCS. Heidelberg: Springer, 2016, 582–603.
[221] Rindal P and Rosulek M. Faster malicious 2-party secure computation with online/offline dual execution. In: Holz T and Savage S (eds.). USENIX Security 2016. USENIX Association, 2016, 297–314.
[222] Shelat A and Shen C-H. Two-output secure computation with malicious adversaries. In: Paterson KG (ed.). EUROCRYPT 2011, volume 6632 of LNCS. Heidelberg: Springer, 2011, 386–405.
[223] Shelat A and Shen C-H. Fast two-party secure computation with minimal assumptions. In: Sadeghi A-R, Gligor VD and Yung M (eds.). ACM CCS 2013. ACM Press, 2013, 523–34.
[224] Wang X, Malozemoff AJ and Katz J. Faster secure two-party computation in the single-execution setting. In: Coron J-S and Nielsen J-B (eds.). EUROCRYPT 2017, Part III, volume 10212 of LNCS. Heidelberg: Springer, 2017, 399–424.
[225] Nielsen JB and Orlandi C. LEGO for two-party secure computation. In: Reingold O (ed.). TCC 2009, volume 5444 of LNCS. Heidelberg: Springer, 2009, 368–86.
[226] Frederiksen TK, Jakobsen TP and Nielsen JB et al. TinyLEGO: An interactive garbling scheme for maliciously secure two-party computation. Cryptology ePrint Archive, Report 2015/309, 2015. http://eprint.iacr.org/2015/309.
[227] Frederiksen TK, Jakobsen TP and Nielsen JB et al. MiniLEGO: Efficient secure two-party computation from general assumptions. In: Johansson T and Nguyen PQ (eds.). EUROCRYPT 2013, volume 7881 of LNCS. Heidelberg: Springer, 2013, 537–56.
[228] Huang Y and Zhu R. Revisiting LEGOs: Optimizations, analysis, and their limit. Cryptology ePrint Archive, Report 2015/1038, 2015. http://eprint.iacr.org/2015/1038.
[229] Kolesnikov V, Nielsen JB and Rosulek M et al. DUPLO: Unifying cut-and-choose for garbled circuits. In: Thuraisingham BM, Evans D, Malkin T and Xu D (eds.). ACM CCS 2017. ACM Press, 2017, 3–20.
[230] Nielsen J B, Schneider T and Trifiletti R. Constant round maliciously secure 2PC with function-independent preprocessing using LEGO. In: NDSS 2017. The Internet Society, 2017.
[231] Zhu R and Huang Y. JIMU: faster LEGO-based secure computation using additive homomorphic hashes. In: Takagi T and Peyrin T (eds.). ASIACRYPT 2017, Part II, volume 10625 of LNCS. Heidelberg: Springer, 2017, 529–72.
[232] Choi SG, Katz J and Malozemoff AJ et al. Efficient three-party computation from cut-and-choose. In: Garay JA and Gennaro R (eds.). CRYPTO 2014, Part II, volume 8617 of LNCS. Heidelberg: Springer, 2014, 513–30.
[233] Lindell Y, Pinkas B and Smart NP et al. Efficient constant round multi-party computation combining BMR and SPDZ. In: Gennaro R and Robshaw MJB (eds.). CRYPTO 2015, Part II, volume 9216 of LNCS. Heidelberg: Springer, 2015, 319–338.
[234] Lindell Y, Smart NP and Soria-Vazquez E. More efficient constant-round multi-party computation from BMR and SHE. In: Hirt M and Smith AD (eds.). TCC 2016-B, Part I, volume 9985 of LNCS. Heidelberg: Springer, 2016, 554–81.
[235] Poddar R, Kalra S and Yanai A et al. Senate: a maliciously-secure MPC platform for collaborative analytics. In: 30th USENIX Security Symposium (USENIX Security 21). USENIX Association, 2021, 2129–46.
[236] Byali M, Joseph A and Patra A et al. Fast secure computation for small population over the Internet. In: Lie D, Mannan M, Backes M and Wang XF (eds.). ACM CCS 2018. ACM Press, 2018, 677–694.
[237] Chandran N, Garay JA and Mohassel P et al. Efficient, constant-round and actively secure MPC: Beyond the three-party case. In: Thuraisingham BM, Evans D, Malkin T, Xu D (eds.). ACM CCS 2017. ACM Press, 2017, 277–294.
[238] Ishai Y, Kumaresan R and Kushilevitz E et al. Secure computation with minimal interaction, revisited. In: Gennaro R and Robshaw MJB (eds.). CRYPTO 2015, Part II, volume 9216 of LNCS. Heidelberg: Springer, 2015, 359–78.
[239] Mohassel P, Rosulek M and Zhang Y. Fast and secure three-party computation: the garbled circuit approach. In: Ray I, Li N and Kruegel C (eds.). ACM CCS 2015. ACM Press, 2015, 591–602.
[240] Byali M, Hazay C and Patra A et al. Fast actively secure five-party computation with security beyond abort. In: Cavallaro L, Kinder J, Wang XF, Katz J (eds.). ACM CCS 2019. ACM Press, 2019, 1573–1590.
[241] Canetti R, Sarkar P and Wang X. Blazing fast OT for three-round UC OT extension. In: Kiayias A, Kohlweiss M, Wallden P and Zikas V (eds.). PKC 2020, Part II, volume 12111 of LNCS. Heidelberg: Springer, 2020, 299–327.

[242] Masny D and Rindal P. Endemic oblivious transfer. In: Cavallaro L, Kinder J, Wang XF and Katz J (eds.). ACM CCS 2019. ACM Press, 2019, 309–326.
[243] McQuoid I, Rosulek M and Roy L. Minimal symmetric PAKE and 1-out-of-N OT from programmable-once public functions. In: Ligatti J, Ou X, Katz J and Vigna G (eds.). ACM CCS 20. ACM Press, 2020, 425–42.
[244] McQuoid I, Rosulek M and Roy L. Batching Base Oblivious Transfers. Cryptology ePrint Archive, Report 2021/682, 2021. https://eprint.iacr.org/2021/682.
[245] Peikert C, Vaikuntanathan V and Waters B. A framework for efficient and composable oblivious transfer. In: Wagner D (ed.). CRYPTO 2008, volume 5157 of LNCS. Heidelberg: Springer, 2008, 554–71.
[246] Chou T and Orlandi C. The simplest protocol for oblivious transfer. In: Progress in Cryptology – LATINCRYPT 2015, volume 9230 of LNCS. Springer International Publishing, 2015, 40–58.
[247] Döttling N, Garg S and Hajiabadi M et al. Two-round oblivious transfer from CDH or LPN. In: Canteaut A and Ishai Y (eds.). EUROCRYPT 2020, Part II, volume 12106 of LNCS. Heidelberg: Springer, 2020, 768–97.
[248] Naor M and Pinkas B. Efficient oblivious transfer protocols. In: Kosaraju SR (ed.). In: 12th SODA. ACM-SIAM, 2001, 448–57.
[249] Branco P, Ding J and Goulão M et al. A framework for universally composable oblivious transfer from one-round key-exchange. In: Albrecht M (ed.). IMA International Conference on Cryptography and Coding – IMACC 2019, volume 11929 of LNCS. Springer International Publishing, 2019, 78–101.
[250] David B and Dowsley R. Efficient composable oblivious transfer from CDH in the global random oracle model. Cryptology ePrint Archive, Report 2020/1291, 2020. https://eprint.iacr.org/2020/1291.
[251] Quach W. UC-secure OT from LWE, Revisited. Cryptology ePrint Archive, Report 2020/819, 2020. https://eprint.iacr.org/2020/819.
[252] Lai YF, Galbraith SD and de Saint Guilhem CD. Compact, Efficient and UC-secure Isogeny-based Oblivious Transfer. Cryptology ePrint Archive, Report 2020/1012. 2020. https://eprint.iacr.org/2020/1012.
[253] Beaver D. Correlated pseudorandomness and the complexity of private computations. In: 28th ACM STOC. ACM Press, 1996, 479–488.
[254] Boyle E, Couteau G and Gilboa N et al. Compressing vector OLE. In: Lie D, Mannan M, Backes M and Wang XF (eds.). ACM CCS 2018. ACM Press, 2018, 896–912.
[255] Blum A, Furst ML and Kearns MJ et al. Cryptographic primitives based on hard learning problems. In: Stinson DR (ed.). CRYPTO'93, volume 773 of LNCS. Heidelberg: Springer, 1994, 278–91.
[256] Boyle E, Couteau G and Gilboa N et al. Correlated pseudorandom functions from variable-density LPN. Cryptology ePrint Archive, Report 2020/1417, 2020. https://eprint.iacr.org/2020/1417.
[257] Goldreich O, Goldwasser S and Micali S. How to construct random functions. J ACM 1986; **33**: 792–807.
[258] Boneh D and Waters B. Constrained pseudorandom functions and their applications. In: Sako K and Sarkar P (eds.). ASIACRYPT 2013, Part II, volume 8270 of LNCS. Heidelberg: Springer, 2013, 280–300.
[259] Kiayias A, Papadopoulos S and Triandopoulos N et al. Delegatable pseudorandom functions and applications. In: Sadeghi A-R, Gligor VD and Yung M (eds.). ACM CCS 2013. ACM Press, 2013, 669–84.
[260] Schoppmann P, Gascón A and Reichert L et al. Distributed vector-OLE: Improved constructions and implementation. In: Cavallaro L, Kinder J, Wang XF and Katz J (eds.). ACM CCS 2019. ACM Press, 2019, 1055–72.
[261] Augot D, Finiasz M and Sendrier N. A fast provably secure cryptographic hash function. Cryptology ePrint Archive, Report 2003/230, 2003. http://eprint.iacr.org/2003/230.
[262] Applebaum B, Damgård I and Ishai Y et al. Secure arithmetic computation with constant computational overhead. In: Katz J, Shacham H (eds.). CRYPTO 2017, Part I, volume 10401 of LNCS. Heidelberg: Springer, 2017, 223–54.
[263] Döttling N, Ghosh S and Nielsen JB et al. TinyOLE: Efficient actively secure two-party computation from oblivious linear function evaluation. In: Thuraisingham BM, Evans D, Malkin T and Xu D (eds.). ACM CCS 2017. ACM Press, 2017, 2263–76.
[264] Ishai Y, Prabhakaran M and Sahai A. Secure arithmetic computation with no honest majority. In: Theory of Cryptography, volume 5444 of LNCS. Berlin, Heidelberg: Springer, 2009. 294–314.
[265] De Castro L, Juvekar C and Vaikuntanathan V. Fast vector oblivious linear evaluation from ring learning with errors. Cryptology ePrint Archive, Report 2020/685, 2020. https://eprint.iacr.org/2020/685.
[266] Baum C, Escudero D and Pedrouzo-Ulloa A et al. Efficient protocols for oblivious linear function evaluation from ring – LWE. In: Galdi C and Kolesnikov V (eds.). SCN 20, volume 12238 of LNCS. Heidelberg: Springer, 2020, 130–49.
[267] Branco P, Döttling N and Mateus P. Two-round oblivious linear evaluation from learning with errors. Cryptology ePrint Archive, Report 2020/635, 2020. https://eprint.iacr.org/2020/635.
[268] Ghosh S, Nielsen JB and Nilges T. Maliciously secure oblivious linear function evaluation with constant overhead. In: Takagi T and Peyrin T (eds.). ASIACRYPT 2017, art I, volume 10624 of LNCS. Heidelberg: Springer, 2017, 629–59.
[269] Chase M, Dodis Y and Ishai Y et al. Reusable non-interactive secure computation. In: Boldyreva A and Micciancio D (eds.). CRYPTO 2019, Part III, volume 11694 of LNCS. Heidelberg: Springer, 2019, 462–488.
[270] Abspoel M, Escudero D and Volgushev N. Secure training of decision trees with continuous attributes. Proc Priv Enhancing Technol 2020; **2021**: 167–87.
[271] Adams S, Choudhary C and De Cock M et al. Privacy-preserving training of tree ensembles over continuous data. Cryptology ePrint Archive, Report 2021/754, 2021. https://eprint.iacr.org/2021/754.
[272] Attrapadung N, Hamada K and Ikarashi D et al. Adam in private: Secure and fast training of deep neural networks with adaptive moment estimation. Cryptology ePrint Archive, Report 2021/736, 2021. https://eprint.iacr.org/2021/736.
[273] Braun L, Demmler D and Schneider T et al. MOTION – A framework for mixed-protocol multi-party computation. Cryptology ePrint Archive, Report 2020/1137, 2020. https://eprint.iacr.org/2020/1137.
[274] Knott B, Venkataraman S and Hannun A et al. CrypTen: secure multi-party computation meets machine learning. In: Proceedings of the NeurIPS Workshop on Privacy-Preserving Machine Learning, 2020.

[275] Nikolaenko V, Weinsberg U and Ioannidis S et al. Privacy-preserving ridge regression on hundreds of millions of records. In: 2013 IEEE Symposium on Security and Privacy. IEEE Computer Society Press, 2013, 334–48.
[276] Wang Q, Ma Q and Li J et al. Enable Dynamic Parameters Combination to Boost Linear Convolutional Neural Network for Sensitive Data Inference. Cryptology ePrint Archive, Report 2020/961, 2020. https://eprint.iacr.org/2020/961.
[277] Liu J, Juuti M and Lu Y et al. Oblivious neural network predictions via MiniONN transformations. In: Thuraisingham BM, Evans D, Malkin T and Xu D (eds.). ACM CCS 2017. ACM Press, 2017, 619–31.
[278] Chandran N, Gupta D and Rastogi A et al. EzPC: Programmable and efficient secure two-party computation for machine learning. In: 2019 IEEE European Symposium on Security and Privacy (EuroS&P), 2019, 496–511.
[279] Simonyan K and Zisserman A. Very Deep Convolutional Networks for Large-scale Image Recognition, 2015. https://arxiv.org/pdf/1409.1556.pdf
[280] Boemer F, Cammarota R and Demmler D et al. MP2ML: A mixed-protocol machine learning framework for private inference. In: Proceedings of the 15th International Conference on Availability, Reliability and Security – ARES'20. ACM, 2020.
[281] Dowlin N, Gilad-Bachrach R and Laine K et al. CryptoNets: Applying neural networks to encrypted data with high throughput and accuracy. In: Proceedings of the 33rd International Conference on International Conference on Machine Learning - ICML'16, 2016, 201–210. https://JMLR.org.
[282] Huang G, Liu Z and Van Der Maaten L et al. Densely connected convolutional networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, 2261–69.
[283] Dalskov A, Escudero D and Keller M. Secure evaluation of quantized neural networks. Proc Priv Enh Technol 2020; **2020**: 355–75.
[284] Howard AG, Zhu M and Chen B et al. MobileNets: Efficient convolutional neural networks for mobile vision applications, 2017.
[285] Spagnolo F, Perri S and Frustaci F et al. Energy-efficient architecture for CNNs inference on heterogeneous FPGA. J Low Power Electron Appl 2020; **10**: 1.
[286] Riazi M S, Weinert C and Tkachenko O et al. Chameleon: a hybrid secure computation framework for machine learning applications. In: Kim J, Ahn G-J, Kim S, Kim Y, López J and Kim T (eds.). ASIACCS 18. ACM Press, 2018, 707–21.
[287] Krizhevsky A, Sutskever I and Hinton G E. Imagenet classification with deep convolutional neural networks. Commun ACM 2017; **60**: 84–90.
[288] Chaudhari H, Choudhury A and Patra A et al. ASTRA: High throughput 3PC over rings with application to secure prediction. In: Proceedings of the 2019 ACM SIGSAC Conference on Cloud Computing Security Workshop – CCSW'19. ACM, 2019, 81–92.
[289] Wagh S, Gupta D and Chandran N. SecureNN: 3-party secure computation for neural network training. Proc Priv Enh Technol 2019; **2019**: 26–49.
[290] Koti N, Pancholi M and Patra A et al. SWIFT: super-fast and robust privacy-preserving machine learning. In: 30th USENIX Security Symposium (USENIX Security 21). USENIX Association, 2021.
[291] He K, Zhang X and Ren S et al. Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, 770–78.
[292] Wagh S, Tople S and Benhamouda F et al. FALCON: Honest-majority Maliciously Secure Framework for Private Deep Learning, 2021. https://arxiv.org/abs/2004.02229.
[293] Byali M, Chaudhari H and Patra A et al. Flash: Fast and robust framework for privacy-preserving machine learning. Cryptology ePrint Archive, Report 2019/1365, 2019. https://eprint.iacr.org/2019/1365.
[294] Koti N, Patra A and Rachuri R et al. Tetrad: Actively Secure 4 PC for Secure Training and Inference. Cryptology ePrint Archive, Report 2021/755, 2021. https://eprint.iacr.org/2021/755.
[295] Carpov S, Deforth K and Gama N et al. Manticore: Efficient framework for scalable secure multiparty computation protocols. Cryptology ePrint Archive, Report 2021/200, 2021. https://eprint.iacr.org/2021/200.
[296] Aly A, Orsini E and Rotaru D et al. Zaphod: Efficiently combining LSSS and garbled circuits in SCALE. In: *Proceedings of the 7th ACM Workshop on Encrypted Computing & Applied Homomorphic Cryptography, WAHC'19*. ACM, 2019, 33–44.
[297] Rotaru D, Smart NP and Tanguy T et al. Actively Secure Setup for SPDZ. Cryptology ePrint Archive, Report 2019/1300, 2019. https://eprint.iacr.org/2019/1300.
[298] Rotaru D and Wood T. MArBled circuits: mixing arithmetic and Boolean circuits with active security. In: Hao F, Ruj S and Sen Gupta S (eds.). INDOCRYPT 2019, volume 11898 of LNCS. Heidelberg: Springer, 2019, 227–49.
[299] Escudero D, Ghosh S and Keller M et al. Improved primitives for MPC over mixed arithmetic-binary circuits. In: Micciancio D and Ristenpart T (eds.). CRYPTO 2020, Part II, volume 12171 of LNCS. Heidelberg: Springer, 2020, 823–852.
[300] Boyle E, Chandran N and Gilboa N et al. Function secret sharing for mixed-mode and fixed-point secure computation. In: Advances in Cryptology – EUROCRYPT 2021, volume 12697 of LNCS. Springer International Publishing, 2021, 871–900.
[301] Boyle E, Gilboa N and Ishai Y. Secure computation with preprocessing via function secret sharing. In: Hofheinz D and Rosen A (eds.). TCC 2019, Part I, volume 11891 of LNCS. Heidelberg: Springer, 2019, 341–371.
[302] ISO/IEC JTC 1/SC 27. ISO/IEC WD 4922-2.3 Information security – Secure multiparty computation – Part 2: Mechanisms based on secret sharing, 2021. https://www.iso.org/standard/80514.html.
[303] National Institute of Standards and Technology (NIST). Multi-party Threshold Cryptography, 2021. https://csrc.nist.gov/Projects/Threshold-Cryptography.
[304] National Institute of Standards and Technology (NIST). Privacy-enhancing Cryptography, 2021. https://csrc.nist.gov/Projects/pec.

Dengguo Feng received his Ph.D. degree in communication engineering and information system from Xidian University in 1995. He is currently a Professor and Ph.D. supervisor. His main research interests include network and information security, trusted computing and information assurance.

Kang Yang received his Ph.D. degree from the Institute of Software, Chinese Academy of Sciences in 2017. He is currently an Associated Professor with the State Key Laboratory of Cryptology, Beijing, China. His research interests include secure multi-party computation, zero-knowledge proofs and so on.