Designs, Codes and Cryptography (2023) 91:1645–1684
https://doi.org/10.1007/s10623-022-01160-x

Maliciously circuit-private multi-key FHE and MPC based on LWE

Nuttapong Attrapadung[1] · Goichiro Hanaoka[1] · Ryo Hiromasa[2] · Takahiro Matsuda[1] · Jacob C. N. Schuldt[1]

Received: 11 October 2021 / Revised: 30 November 2022 / Accepted: 2 December 2022 /
Published online: 30 December 2022

Abstract
In this paper, we construct multi-key homomorphic and fully homomorphic encryption (resp. MKHE and MKFHE) schemes with malicious circuit privacy. Our schemes are based on learning with errors (LWE) besides appropriate circular security assumptions. In contrast, the previous maliciously circuit-private MKFHE scheme by Chongchitmate and Ostrovsky (PKC, 2017) is based on the non-standard decisional small polynomial ratio (DSPR) assumption with a super-polynomial modulus, besides ring learning with errors and circular security assumptions. We note that it was shown by Albrecht et al. (CRYPTO, 2016) that there exists a sub-exponential time attack against this type of DSPR assumption. The main building block of our maliciously circuit-private MKFHE scheme is a (plain) MKFHE scheme by Brakerski et al. (TCC, 2017), and the security of our schemes is proven under the hardness of LWE with sub-exponential modulus-to-noise ratio and circular security assumptions related to the Brakerski et al. scheme. Furthermore, based on our MKFHE schemes, we construct four-round multi-party computation (MPC) protocols with circuit privacy against a semi-honest

Communicated by D. Stehle.

N. Attrapadung, G. Hanaoka, R. Hiromasa, T. Matsuda, and J. C. N. Schuldt have contributed equally to this work.

✉ Ryo Hiromasa
Hiromasa.Ryo@aj.MitsubishiElectric.co.jp

Nuttapong Attrapadung
n.attrapadung@aist.go.jp

Goichiro Hanaoka
hanaoka-goichiro@aist.go.jp

Takahiro Matsuda
t-matsuda@aist.go.jp

Jacob C. N. Schuldt
jacob.schuldt@aist.go.jp

[1] National Institute of Advanced Industrial Science and Technology, 2-3-26, Aomi, Koto-ku, Tokyo 135-0064, Japan

[2] Mitsubishi Electric, 5-1-1, Ofuna, Kamakura, Kanagawa 247-0056, Japan

server and malicious clients in the plain model. The protocols are obtained by combining our schemes with a maliciously sender-private oblivious transfer protocol and a circuit garbling scheme, all of which can be instantiated only assuming LWE.

Keywords Multi-key fully homomorphic encryption · Circuit privacy · Learning with errors · Multi-party computation

Mathematics Subject Classification 94A60

1 Introduction

Fully homomorphic encryption (FHE) [8, 11, 12, 22, 23, 40] is a special type of public key encryption that allows functions to be evaluated over encrypted data. FHE is a powerful primitive often used as a building block in other schemes and protocols. A typical application of FHE is the construction of a multi-party computation (MPC) protocol between two semi-honest parties, which allows the parties to jointly perform a computation over their inputs, while revealing nothing more than the output of the computation. Specifically, the MPC protocol can be constructed as follows. The first party P_1 encrypts his input x under his own public key and sends the ciphertext c to the second party P_2. Then P_2 evaluates a function f homomorphically on c and his own input y, and sends the result back to P_1, who decrypts the final ciphertext to obtain $f(x, y)$. An important property of this protocol is that the communication complexity is independent of the complexity of the function f being evaluated. Multi-key fully homomorphic encryption (MKFHE) [2, 10, 18, 31, 35] is a variant of FHE which allows evaluation of functions over ciphertexts generated under different public keys. Similar to FHE leading to a natural construction of a two-party computation protocol, MKFHE can be used as a basis for a multi-party computation protocol, where the parties encrypt their input under their own public key, then homomorphically evaluate the relevant function over all ciphertexts, and collaboratively run a multiparty decryption protocol on the evaluated ciphertext.

Client-Server MPC. In this paper, we consider a special type of MPC protocols called *client-server* MPC protocols in which the parties play different roles, namely server and client. The single server is given a description of a function to be computed in the protocol, and clients are given inputs to the function. Such protocols have been considered in the literature of server-aided MPC [29, 31] and circuit-private homomorphic encryption [17, 28, 36]. (The formal definition of the client-server MPC protocol can be found on Definition 5.1.) In client-server MPC, we can consider two types of security notions: *client privacy* and *circuit privacy* [1]. Client privacy requires that the clients' inputs are kept semantically secure, whereas circuit privacy requires that no information about the function computed in the MPC protocol is revealed beyond the output with respect to the clients' inputs. In this paper, we consider circuit privacy against malicious adversaries [28], namely even if the server is given malicious inputs (such as ill-formed ciphertexts or public keys) from clients, the output of the MPC protocol does not reveal any information about the computed function.

FHE and MPC with malicious circuit privacy. In [17], Chongchitmate and Ostrovsky showed a three-round MPC protocol with circuit privacy. Their protocol is secure against a semi-honest server and malicious clients in the plain model under the (non-standard) decisional small polynomial ratio (DSPR) assumption with super-polynomial modulus (in addition to learning with errors over a polynomial ring and appropriate circular security

[1] This is referred to as server privacy in [28], but here we use the term "circuit privacy" to relate the notion to circuit privacy of FHE.

assumptions for the underlying building blocks). We note that it was shown by Albrecht et al. [1] that there exists a sub-exponential time attack against this type of DSPR assumption. This gives rise to the following natural question:

Can a client-server MPC protocol with (malicious) circuit privacy be obtained from standard assumptions?

The main building block of the protocol in [17] is a MKFHE scheme with circuit privacy against malicious adversaries. This scheme is in turn constructed based on the MKFHE scheme by Lopez-Alt et al. [31], which leads to the dependency on the DSPR assumption, as the scheme from [31] requires this.

In general, a client-server MPC protocol with circuit privacy against malicious clients can be constructed from any maliciously circuit-private MKFHE scheme by combining this with a statistically sender-private oblivious transfer (OT) protocol and information-theoretic randomized encoding [27]. Such an OT protocol exists under standard assumptions, e.g. the LWE-based protocol [9], so the above question can be rephrased as the following question, which was also raised in [17] as an open problem:

Can a maliciously circuit-private MKFHE scheme be obtained from standard assumptions?

1.1 Our results

In this paper, we answer the above questions in the affirmative, with the only caveat being that appropriate circular security assumptions are additionally required (note that all of the currently known (non-leveled) FHE schemes not based on obfuscation, require circular security assumptions). Concretely, we firstly construct a maliciously circuit-private MKHE [2] scheme for branching programs based on LWE as well as weak circular security of the (plain) MKFHE scheme by Brakerski et al. [15], which is one of the main building blocks of our scheme. Informally, we prove the following theorem:

Theorem 1.1 (Informal) *Assuming the hardness of LWE with sub-exponential modulus-to-noise ratio (and weak circular security of [15]), there exists a maliciously circuit-private MKHE scheme with distributed setup for branching programs.*

Secondly, we show how the obtained scheme can be combined with a plain MKFHE scheme (such as [15]), to obtain a fully homomorphic maliciously circuit-private scheme. To obtain this result, we rely on a somewhat non-standard circular security assumption. We note that a similar assumption is required to show the security of the previous construction of a maliciously circuit-private MKFHE in [17]. (The details of the circular security assumption are discussed in Sect. 4.3.) Informally, we show the following result.

Theorem 1.2 (Informal) *Assuming the existence of a distributed-setup maliciously circuit-private MKHE scheme for branching programs and a distributed-setup (plain) MKFHE scheme which are "jointly circular secure", there exists a maliciously circuit-private MKFHE scheme with distributed setup.*

Based on either of our MK(F)HE schemes, we can build a four-round MPC protocol with circuit privacy against a semi-honest server and malicious clients in the plain model. All the additional building blocks required for our protocol can be instantiated assuming only LWE,

[2] Here we purposely write "MKHE" (without "F") since it does not support all circuits.

Table 1 A comparison among MPC protocol paradigms for circuits

MPC for circuits	Main tools	Communication complexity independent of circuit size?	Can client computation be independent of circuit size?
GMW/BGW paradigm e.g., [6, 7, 19, 24]	secret sharing	×	×
Yao/BMR paradigm e.g., [5, 26, 30, 41]	garbled circuit	×	×
FHE based paradigm e.g., [31, 35] (and this work)	multi-key FHE	✓	✓

and hence, from our circuit-private MKHE scheme for branching programs (Theorem 1.1), we obtain the following:

Theorem 1.3 (Informal) *Assuming the hardness of LWE with sub-exponential modulus-to-noise ratio (and weak circular security of [15]), there exists a four-round client-server MPC protocol for branching programs (NC^1 circuits) with circuit privacy against a malicious adversary corrupting only clients in the plain model.*

Furthermore, based on our MKFHE scheme (Theorem 1.2), we obtain:

Theorem 1.4 (Informal) *Assuming the existence of a distributed-setup maliciously circuit-private MKHE scheme for branching programs and a distributed-setup (plain) MKFHE scheme which are jointly circular secure, there exists a four-round client-server MPC protocol with circuit privacy against a malicious adversary corrupting only clients in the plain model.*

Comparison with Existing MPC Protocols. We first loosely compare MKFHE-based MPC protocols for circuits with other well-known general paradigms, namely, secret-sharing-based and garbled-circuit-based MPC in Table 1 .[3] [4] MPC protocols based on the latter two paradigms generally do not aim at hiding the evaluated circuit. While it is in principle possible to hide the *structure* of the circuit, as we highlight below in our discussion of a related primitive called private function evaluation (PFE), information regarding the *circuit size* leaks as the communication complexity depends on this. In contrast, MKFHE-based MPC protocols makes the communication complexity and also client computation independent of the circuit size.

We next compare our MPC results among existing MK-FHE based client-server MPC protocols in the plain model in Table 2. We remark that these protocols achieve communication complexity and client computation independent of the circuit being evaluated. The main feature of our protocols is that they can be based on more standard computational assumptions (LWE), while the previous two works [17, 31] make use of the non-standard DSPR assumptions. Note that besides the computational assumptions listed, appropriate circular security assumptions are needed in all of the listed schemes. As a trade-off, our protocols require one more round compared to [17]; intuitively, this is due to our design where participating parties have to share individual public parameters used to generate public and secret keys for the underlying maliciously circuit-private MKFHE scheme.

[3] Note that there are also combinations of these paradigms such as ABY/ABY3 [20, 32] which combine secret-sharing-based and garbled-circuit-based MPC; these protocols inherit the circuit size dependency properties (the last two columns in Table 1) from their components.

[4] Note that in Table 1, we compare general MPC, not only client-server MPC.

Table 2 Comparison among existing MKFHE-based client-server MPC protocols in the plain model. Besides the computational assumptions listed, appropriate circular security assumptions are needed in all of the listed schemes. Client privacy refers to privacy of honest clients' input against corrupted clients

Construction	Circuit class	Rounds	Assumptions	Circuit Privacy	Client Privacy
[31]	Any	5	DSPR, RLWE	No	Semi-honest
[17]	Any	3	DSPR, RLWE	Yes	Malicious
Ours 1	NC^1	4	LWE	Yes	Malicious
Ours 2	Any	4	LWE	Yes	Malicious

Relation to private function evaluation Private function evaluation (PFE) is a special type of MPC which allows a function, held by one party, to be computed while being kept secret. Such a protocol can be achieved by computing a universal circuit in an ordinary MPC protocol, and treating the (description of the) function as a client input. While the universal circuit approach hides the function itself, it does not guarantee that the size of the evaluated function is kept secret. In contrast, our MPC protocol provides this guarantee and achieves a communication complexity that is independent of the size of the evaluated function due to the use of MKFHE. Furthermore, the use of universal circuits incurs a significant overhead in terms of computation. A (boolean or arithmetic) circuit of size g requires a universal circuit of size at least $O(g \log g)$ [37, 39] to be computed. A more practical approach to PFE that does not rely on a universal circuit is considered in [33, 34] (in particular, [34] constructs an actively secure PFE which, unlike the security considered in this paper, allows corrupting the function holder), but the complexity of their protocols still depends on the size of the evaluated circuit.

1.2 Overview of our constructions

Here, we give an overview of the construction techniques we use to obtain our maliciously circuit-private MK(F)HE schemes. As our basic approach is similar to that of [17], we start by recalling the ideas behind the circuit-private MKFHE construction from [17].

Prior approach to circuit-private MKFHE Chongchitmate and Ostrovsky [17] constructed the first (maliciously) circuit-private MKFHE scheme. The scheme is built on a MKFHE scheme with a special property called *private expandability*. To homomorphically evaluate a function over ciphertexts, a MKFHE scheme usually transforms an input ciphertext ct_i (generated under a single public key pk_i) into a ciphertext $\widetilde{\mathsf{ct}}$ that relates to all the keys $\{\mathsf{pk}_i\}_i$ involved in the homomorphic evaluation. Private expandability guarantees that no efficient algorithm can distinguish $\widetilde{\mathsf{ct}}$, from an expanded ciphertext $\widetilde{\mathsf{ct}}'$ obtained by expanding a ciphertext ct_j under a different public key pk_j for any $j \neq i$. The authors constructed a privately-expandable scheme from the MKFHE scheme of [31]. The private expansion algorithm of the scheme uses the noise smudging technique of [14, 21, 25] to remove the dependency on the key pk_i under which the original ciphertext ct_i is constructed. Private expandability leads to a semi-honest circuit-private MKHE scheme for branching programs by using the technique of [28]. Intuitively, private expandability hides by which key the expanded input was originally encrypted. The semi-honest circuit-private scheme for branching programs is transformed into a maliciously circuit-private scheme by using the maliciously circuit-private single-key FHE scheme of [36] to homomorphically check whether all pk_i's and ct_i's are well-formed.

Finally, a fully homomorphic scheme is obtained by combining the maliciously circuit-private scheme with a normal MKFHE scheme with decryption in NC^1. It should be noted that all of the above schemes [17] (and technique of [36]) are in the plain model.

Privately-expandable MKFHE with distributed setup The definitions and constructions in [17] crucially rely on the use of a setup-free MKFHE scheme to obtain a scheme in the plain model. This eventually leads to their instantiation being based on the non-standard DSPR assumption (via [31]). We take a different approach. Specifically, we construct a maliciously circuit-private MKFHE scheme *with distributed setup*, in which every party P_i independently generates a public parameter pp_i, and $\{\mathsf{pp}_i\}_i$ are used to generate keys and ciphertexts. This will allow us to obtain a scheme based on standard assumptions, without having to introduce a common reference string (CRS) or trusted setup. The concept of a distributed setup for a MKFHE scheme is not new, and Brakerski, Halevi, and Polychroniadou [15] proposed a scheme with this property for the purpose of obtaining a four-round secure MPC protocol in the plain model. Note, however, that the results from [15] cannot be used directly in our approach, as the concept of private expandability was not considered in [15]. To address this, we firstly introduce a generalization of private expandability to MKFHE with distributed setup, in which the ciphertext expansion algorithm Expand takes as input $\{\mathsf{pp}_i\}_i$ generated by the P_i's. We then proceed to prove that the scheme of [15] can in fact be extended to yield a scheme providing our notion of private expandability. To prove this, we use the noise smudging technique, which incurs the sub-exponential modulus-to-noise ratio in the LWE assumption. As a result, we obtain a privately-expandable MKFHE scheme based on LWE without having to introduce a CRS or trusted setup.

Achieving malicious circuit privacy. To obtain a maliciously circuit-private MKFHE scheme, we extend the approach taken by [17] to our setting. Specifically, we firstly define malicious circuit privacy for MK(F)HE with distributed setup, in which $\{\mathsf{pp}_i\}_i$ generated in the distributed setup are given to the homomorphic evaluation algorithm. We then construct a maliciously circuit-private scheme using a privately-expandable MKFHE scheme with distributed setup. More precisely, private expandability allows us to construct a semi-honest circuit-private MKHE *for branching programs* in a similar way to [17, 28]. To achieve malicious circuit privacy, we use an additional (maliciously) circuit-private *single-key* FHE scheme to homomorphically check whether inputs to the homomorphic evaluation algorithm are well-formed. Note that for this to work, it must be possible for the evaluation algorithm to check the validity of the public parameters generated by the distributed setup procedure (without access to the randomness used to generate these). However, for our particular privately-expandable MKFHE scheme, all values pp_i are picked uniformly at random from the appropriate domain, and hence we can check the validity of the pp_i's simply via a membership test. Furthermore, similar to [17], the construction requires public keys to contain bit-wise encryptions of private keys of the privately-expandable MKFHE scheme, and hence requires the latter to be weakly circular secure. Lastly, note that the required maliciously circuit-private single-key FHE scheme can be constructed from an information-theoretic randomized encoding scheme, an OT protocol, and a single-key FHE scheme, as shown by Ostrovsky et al. [36]. Since the latter two primitives can be instantiated assuming only LWE [8, 9, 12], we obtain a maliciously circuit-private scheme for branching programs based on LWE (besides the required circular assumption).

Finally, we construct a fully homomorphic scheme by combining the maliciously circuit-private scheme for branching programs and a standard MKFHE scheme (such as the fully homomorphic variant of [15] based on LWE). Inputs are encrypted by the standard MKFHE scheme, and homomorphic evaluation is done via the evaluation algorithm of the standard

MKFHE scheme, while the validity of the inputs is checked using our maliciously circuit-private scheme for branching programs. Furthermore, since the key generation and encryption of the MKFHE scheme from [15] are in NC^1, the maliciously circuit-private scheme for branching programs can homomorphically evaluate these to check whether the inputs to the homomorphic evaluation are generated properly. As a result, we obtain a maliciously circuit-private MKFHE scheme with distributed setup. We emphasize that, compared to the scheme from [17], the computational assumption required for our scheme is significantly weakened, from the non-standard DSPR assumption to the LWE assumption.

MPC protocol with malicious circuit privacy based on LWE. In a similar fashion to [17], we show that we can construct a MPC protocol with malicious circuit privacy based on our circuit-private MKFHE scheme combined with OT secure against malicious receivers and circuit garbling. However, as our scheme has a distributed setup procedure, we need one additional round in our MPC protocol for sharing the public parameters generated by the clients. Note that the observation made in [3, 15] is equally valid for our protocol: as the public parameters shared by clients in the first round are just uniformly random strings, the server need not check whether the parameters shared by malicious clients are properly generated. This allows us to obtain a client-server MPC protocol with malicious circuit privacy. We note that the additional building blocks required in our protocol (OT and circuit gabling) are known to exist under LWE [9, 41].

1.3 Organization

The rest of this paper is organized as follows. Section 2 introduces mathematical preliminaries and definitions of cryptographic primitives for this paper. In Sect. 3, we show how to construct a privately-expandable MKFHE scheme with distributed setup under LWE with sub-exponential modulus-to-noise ratio and the circular security of the underlying MKFHE scheme [15]. In Sect. 4, we first construct a (maliciously) circuit-private MKHE with distributed setup for branching programs from the privately-expandable scheme, and then build up a fully homomorphic scheme via a LWE-based MKFHE scheme with distributed setup. Section 5 shows our four-round MPC protocols based on our MK(F)HE schemes with distributed setup.

2 Preliminaries

In this section, we review notations, mathematical preliminaries, and definitions for homomorphic encryption.

Notations We denote the set of natural numbers, integers, and real numbers by $\mathbb{N}$, $\mathbb{Z}$, and $\mathbb{R}$, respectively. For $d \in \mathbb{N}$, we represent $\{1, 2, \ldots, d\}$ by $[d]$. For a countable set S, $a \xleftarrow{\$} S$ denotes that $a \in S$ is chosen uniformly at random from S. For a probability distribution $\mathcal{P}$ (over some set), $[\mathcal{P}]$ denotes the support of $\mathcal{P}$ (i.e. $\{x : \Pr[\mathcal{P} = x] > 0\}$), and $b \xleftarrow{\$} \mathcal{P}$ denotes that b is sampled according to $\mathcal{P}$. $\mathsf{negl}(\lambda)$ represents an unspecified negligible function. Let $X = \{X_\lambda\}_{\lambda\in\mathbb{N}}$ and $Y = \{Y_\lambda\}_{\lambda\in\mathbb{N}}$ be ensembles of random variables, and $\Delta(X, Y)$ be the statistical distance between them. For $\epsilon > 0$, we say that the two ensembles of random variables, X and Y, are ϵ-close if $\Delta(X, Y) \leq \epsilon$, and X and Y are statistically indistinguishable if they are ϵ-close for $\epsilon = \mathsf{negl}(\lambda)$, and denote it by $X \approx_s Y$. Also we use $X \approx_c Y$ to mean that X and Y are computationally indistinguishable, i.e., $|\Pr[\mathcal{D}(X_\lambda) \to 1] - \Pr[\mathcal{D}(Y_\lambda) \to$

$1]| = \mathsf{negl}(\lambda)$ for any probabilistic polynomial-time (PPT) Turing machine $\mathcal{D}$. For simplicity, we sometimes abuse notation and write $X_\lambda \approx_s Y_\lambda$ (instead of $X \approx_s Y$), and we do the same for $\approx_c$.

Vectors are in column form and written by bold lower-case letters (e.g., $\mathbf{x}$). The i-th element of the vector $\mathbf{x}$ is represented by x_i. We denote the ℓ_∞ norm (max norm) of the vector $\mathbf{x}$ by $\|\mathbf{x}\|$. The inner-product of two vectors $\mathbf{x}$ and $\mathbf{y}$ is written as $\langle \mathbf{x}, \mathbf{y} \rangle$. We denote matrices as bold capital letters (e.g., $\mathbf{X}$) and the (i, j)-th element of the matrix $\mathbf{X}$ is represented by $\mathbf{X}[i, j]$. For a matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$, the ℓ_∞ norm of $\mathbf{X}$ is defined as $\|\mathbf{X}\| := \max_{i \in [n], j \in [m]} \{\|\mathbf{X}[i, j]\|\}$. The notation $\mathbf{X}^T \in \mathbb{Z}^{n \times m}$ represents the transpose of $\mathbf{X}$. For two matrices $\mathbf{A} \in \mathbb{Z}^{m \times n_1}$ and $\mathbf{B} \in \mathbb{Z}^{m \times n_2}$, $[\mathbf{A} \| \mathbf{B}] \in \mathbb{Z}^{m \times (n_1 + n_2)}$ is the matrix obtained by concatenating $\mathbf{A}$ and $\mathbf{B}$. $\mathbf{I}_n$ denotes the $n \times n$ identity matrix, and $\mathbf{0}_{n \times m}$ denotes the $n \times m$ matrix all of whose entries are 0. For any $i \in [n]$, $\mathbf{u}_i \in \{0, 1\}^n$ represents the i-th standard basis vector of dimension n.

2.1 Gaussian, learning with errors, and gadget matrix

Gaussian For any real $\sigma > 0$, a Gaussian function on $\mathbb{R}^n$ centered around $\mathbf{0}$ with parameter σ is defined as $\rho_\sigma(\mathbf{x}) := \exp(-\pi \cdot \|\mathbf{x}\|^2 / \sigma^2)$ for all $\mathbf{x} \in \mathbb{R}^n$. The (discrete) Gaussian distribution over $\mathbb{Z}^n$ of parameter σ, denoted by D_σ, is defined to be the distribution with the probability density function $D_\sigma(\mathbf{x}) := \rho_\sigma(\mathbf{x}) / \sum_{\mathbf{x} \in \mathbb{Z}^n} \rho_\sigma(\mathbf{x})$ for all $\mathbf{x} \in \mathbb{Z}^n$.

We recall the *noise smudging* (or so-called noise flooding) lemma. It states that adding a noise according to the Gaussian distribution with parameter superpolynomial in λ vanishes dependence on the original distribution.

Lemma 2.1 (Gaussian Noise Smudging [14, 25]) *Let $\sigma' > 0$ and $y \in \mathbb{Z}$ be arbitrary. Then, $\Delta(D_{\sigma'}, D_{\sigma'} + y) \leq \frac{\|y\|}{\sigma'}$.*

Learning with errors The *learning with errors (LWE)* problem was first introduced by Regev [38]. We will base the security of our construction on the hardness of the decisional version of the problem, called the decisional LWE problem. For positive integers n and $q \geq 2$, let $A_{\mathbf{s}, D}$ be the distribution obtained by choosing a vector $\mathbf{a} \xleftarrow{\$} \mathbb{Z}_q^n$ uniformly at random and noise term $e \xleftarrow{\$} D$, and outputting $(\mathbf{a}, \mathbf{s}^T \mathbf{a} + e \bmod q) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$. The (decisional) LWE problem is to distinguish $A_{\mathbf{s}, D}$ where $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$, from the uniform distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_q$. The LWE assumption states that this problem is intractable for any PPT algorithm. In this paper, we will use the LWE assumption in which q is superpolynomial in λ, and D is a Gaussian distribution whose samples have norm bounded by some $B \in \mathbb{N}$ polynomial in λ with overwhelming probability.

Gadget matrix Let $\mathbf{g}^T := (1, 2, \ldots, 2^{\lceil \log q \rceil - 1})$ be the vector consisting of the powers of 2. For $n \in \mathbb{N}$, we define the special matrix $\mathbf{G} \in \mathbb{Z}_q^{n \times n \cdot (\lceil \log q \rceil)}$ (called the gadget matrix) that has the vector $\mathbf{g}$ in diagonal and 0 in other elements, namely

$$\mathbf{G} := \begin{bmatrix} -\mathbf{g}^T- & & \\ & \ddots & \\ & & -\mathbf{g}^T- \end{bmatrix} \in \mathbb{Z}_q^{n \times n \cdot (\lceil \log q \rceil)}.$$

Let $\mathbf{G}^{-1} : \mathbb{Z}_q^n \to \{0, 1\}^{n \cdot (\lceil \log q \rceil)}$ be the operation such that $\mathbf{G} \cdot \mathbf{G}^{-1}(\mathbf{x}) = \mathbf{x}$ for any vector $\mathbf{x} \in \mathbb{Z}_q^n$. Such operation can be obtained by decomposing every element of the given vector $\mathbf{x}$ in binary representation. We will also abuse the notation and allow $\mathbf{G}^{-1}$ to take a matrix

$\mathbf{M} \in \mathbb{Z}^{n \times k}$ as input, apply $\mathbf{G}^{-1}$ to each column vector of $\mathbf{M}$, and output the $(n \cdot (\lceil \log q \rceil))$-by-$k$ matrix consisting of the (horizontal) concatenation of all of the outputs, for any $k \in \mathbb{N}$. Then, for a matrix $\mathbf{M} \in \mathbb{Z}^{n \times k}$, it holds that $\mathbf{G} \cdot \mathbf{G}^{-1}(\mathbf{M}) = \mathbf{M}$.

2.2 Branching programs

Here, we recall the definition of branching programs. (The definition here is mostly taken verbatim from [17].)

Definition 2.1 *(Branching program)* A (binary) branching program P over $\{0, 1\}^n$ consists of a tuple $(G = (V, E), v_0, T, \phi_V, \phi_E)$ with the following properties:

- G is a connected directed acyclic graph. We denote by $\Gamma(v)$ the set of the child nodes of $v \in V$.
- v_0 is the initial node with indegree 0.
- $T \subseteq V$ is the set of terminal nodes of outdegree 0. Any node in $V \backslash T$ has outdegree 2.
- $\phi_V : V \to [n] \cup \{0, 1\}$ is the node-labeling function with $\phi_V(v) \in \{0, 1\}$ for $v \in T$, and $\phi_V(v) \in [n]$ for $v \in V \backslash T$.
- $\phi_E : E \to \{0, 1\}$ is the edge-labeling function, such that the outgoing edges from each vertex are labeled by different values.

The *height* of $v \in V$, denoted $\mathsf{height}(v)$, is the length of the longest path from v to a node in T. The *length* of P is the height of v_0. The *width* of P is the maximum number of vertices with the same height.

On input $x \in \{0, 1\}^n$, $P(x)$ is defined as follows: Follow the path induced by x from v_0 to a node $v_\ell \in T$, where an edge (v, v') is in the path if $x_{\phi_V(v)} = \phi_E(v, v')$. (By the property of ϕ_E, such v' is unique.) Then $P(x) := \phi_V(v_\ell)$. Similarly, we also define $P_v(x)$ by following the path from any node $v \in V$ instead of v_0.

We say that a branching program $P = (G = (V, E), v_0, T, \phi_V, \phi_E)$ is *layered* if for any $e = (v, v') \in E$, we have $\mathsf{height}(v) = \mathsf{height}(v') + 1$.

By Barrington's theorem [4], all languages in NC^1 can be computed by a poly-sized layered branching program with constant width. Since the decryption circuit of a number of existing LWE-based FHE schemes (e.g., [13, 15, 23, 35]) is in NC^1, there exists a layered branching program of polynomial length that computes the decryption circuit of such LWE-based FHE schemes.

2.3 Single-key homomorphic encryption and its circuit privacy

Here, we recall the definitions for single-key HE.

Definition 2.2 *(Single-key Homomorphic Encryption)* A single-key HE scheme SKHE for a class of circuits $\mathcal{C}$ consists of the four algorithms $(\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ with the following properties:

- (Syntax)
  - $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KG}(1^\lambda)$: This is the key generation algorithm that takes a security parameter 1^λ as input, and outputs a public/secret key pair $(\mathsf{pk}, \mathsf{sk})$.
  - $\mathsf{ct} \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}, x)$: This is the encryption algorithm that takes a public key pk and a plaintext $x \in \{0, 1\}$ as input, and outputs a ciphertext ct.

- $\widehat{\mathsf{ct}} \xleftarrow{\$} \mathsf{Eval}(C, \mathsf{pk}, (\mathsf{ct}_k)_{k\in[n]})$: This is the homomorphic evaluation algorithm that takes a circuit $C \in \mathcal{C}$ (with domain $\{0,1\}^n$), a public key pk, and n ciphertexts $(\mathsf{ct}_k)_{k\in[n]}$ as input, and outputs an evaluated ciphertext $\widehat{\mathsf{ct}}$.
- $\widehat{x} := \mathsf{Dec}(\mathsf{sk}, \widehat{\mathsf{ct}})$: This is the (deterministic) decryption algorithm that takes a secret key sk and an evaluated ciphertext $\widehat{\mathsf{ct}}$ as input, and outputs a decryption result $\widehat{x}$.

- (Correctness) For any circuit $C \in \mathcal{C}$ (with n-bit input), and any $(x_1, \ldots, x_n) \in \{0,1\}^n$, if $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KG}(1^\lambda)$, $\mathsf{ct}_k \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}, x_k)$ for every $k \in [n]$, and $\widehat{\mathsf{ct}} \xleftarrow{\$} \mathsf{Eval}(C, \mathsf{pk}, (\mathsf{ct}_k)_{k\in[n]})$, then we have $\Pr[\mathsf{Dec}(\mathsf{sk}, \widehat{\mathsf{ct}}) \neq C(x_1, \ldots, x_n)] = \mathsf{negl}(\lambda)$.
- (Semantic Security) Defined in the same way as that for ordinary public-key encryption.

We say that SKHE is *weakly circular secure* if it remains secure even if bit-wise encryptions of a secret key are attached to a public key. We say that SKHE is *fully homomorphic* if $\mathcal{C}$ is a set of all circuits of polynomial size.

Circuit privacy Here, we recall the formal definition of malicious circuit privacy for single-key HE [36].

Definition 2.3 *(Malicious Circuit Privacy for Single-Key HE [36])* Let $\mathsf{SKHE} = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ be a single-key HE for a circuit class $\mathcal{C}$. We say that SKHE is *maliciously circuit private* if there exist an unbounded algorithm Sim (called the simulator) and an unbounded deterministic algorithm Ext (called the extractor) such that for all circuits $C \in \mathcal{C}$ (with n-bit input), and all possibly malformed public keys pk^* and ciphertexts $\mathsf{ct}_1^*, \ldots, \mathsf{ct}_n^*$, we have

$$\mathsf{Eval}(C, \mathsf{pk}^*, (\mathsf{ct}_k^*)_{k\in[n]}) \approx_s \mathsf{Sim}(\mathsf{pk}^*, (\mathsf{ct}_k^*)_{k\in[n]}, C(x_1^*, \ldots, x_n^*)),$$

where $x_k^* := \mathsf{Ext}(\mathsf{pk}^*, \mathsf{ct}_k^*)$ for all $k \in [n]$.

2.4 Oblivious transfer

Here, we recall the definition of a statistically sender-private two-message oblivious transfer (OT) protocol against malicious receivers, which is known to exist under the LWE assumption due to Brakerski and Döttling [9].

Definition 2.4 A two-message OT protocol OT consists of the three algorithms $(\mathsf{Q}, \mathsf{A}, \mathsf{D})$ with the following properties:

- (Syntax)
  - $(q, \mathsf{st}) \xleftarrow{\$} \mathsf{Q}(1^\lambda, b)$: This is the receiver's first algorithm that takes a security parameter 1^λ and a selection bit $b \in \{0,1\}$ as input, and outputs a receiver message q and a secret state st.
  - $a \xleftarrow{\$} \mathsf{A}(q, s_0, s_1)$: This is the sender's algorithm that takes a receiver message q and two strings s_0, s_1 as input, and outputs a sender message a.
  - $s' := \mathsf{D}(a, \mathsf{st})$: This is the (deterministic) receiver's second algorithm that takes a sender message a and a secret state st as input, and outputs a string s'.
- (Correctness) For all $\lambda \in \mathbb{N}$, $b \in \{0,1\}$, and $s_0, s_1 \in \{0,1\}^*$ such that $\|s_0\| = \|s_1\|$, if $(q, \mathsf{st}) \xleftarrow{\$} \mathsf{Q}(1^\lambda, b)$ and $a \xleftarrow{\$} \mathsf{A}(q, s_0, s_1)$, then it holds that $s_b = \mathsf{D}(a, \mathsf{st})$.
- (Receiver Privacy) It holds that $\mathsf{Q}(1^\lambda, 0) \approx_c \mathsf{Q}(1^\lambda, 1)$.

- (Statistical Sender Privacy against Malicious Receivers) There exist possibly computationally unbounded algorithms Ext (called the *extractor*) and Sim (called the *simulator*) such that for any receiver message q (which could be outside the image of Q) and inputs (s_0, s_1) with $\|s_0\| = \|s_1\|$, we have $\mathsf{A}(q, s_0, s_1) \approx_s \mathsf{Sim}(q, s_{b^*})$, where $b^* := \mathsf{Ext}(q)$.

2.5 Circuit garbling

Here, we recall the definition of a circuit garbling scheme. In our MPC construction, we will use a *projective* circuit garbling scheme whose encoding is a list of tokens, one pair for each bit of an input $x \in \{0,1\}^n$ for a circuit being garbled. Such a garbling scheme can be realized from any one-way function, and thus under the LWE assumption.

Definition 2.5 A (projective) circuit garbling scheme GC is a pair of the algorithms $(\mathsf{GCircuit}, \mathsf{GEval})$ with the following properties:

- (Syntax)
  - $(G, e) \xleftarrow{\$} \mathsf{GCircuit}$: This is the garbling algorithm that takes a security parameter 1^λ and a circuit C with n-bit input (for some polynomial $n = n(\lambda)$) as input, and outputs a garbled circuit G and a set of tokens $e = (X_i^0, X_i^1)_{i\in[n]}$.
  - $y = \mathsf{GEval}(G, (X_i)_{i\in[n]})$: This is the (deterministic) evaluation algorithm that takes a garbled circuit G and n tokens $(X_i)_{i\in[n]}$ as input, and outputs some value y.
- (Correctness) For any $\lambda, n \in \mathbb{N}$, any n-bit input circuit C, and any $x = (x_1, \ldots, x_n) \in \{0,1\}^n$, if $(G, e = (X_i^0, X_i^1)_{i\in[n]}) \xleftarrow{\$} \mathsf{GCircuit}(1^\lambda, C)$, then we have $\mathsf{GEval}(G, (X_i^{x_i})_{i\in[n]}) = C(x)$.
- (Security) For any two circuits C_0, C_1 (with n-bit input) of the same size and any two inputs $x_0 = (x_{1,0}, \ldots, x_{n,0}), x_1 = (x_{1,1}, \ldots, x_{n,1}) \in \{0,1\}^n$ such that $C_0(x_0) = C_1(x_1)$, if $(G_b, e_b = (X_{i,b}^0, X_{i,b}^1)_{i\in[n]}) \xleftarrow{\$} \mathsf{GCircuit}(1^\lambda, C_i)$ for both $b \in \{0,1\}$, then we have $(G_0, (X_{i,0}^{x_{i,0}})_{i\in[n]}) \approx_c (G_1, (X_{i,1}^{x_{i,1}})_{i\in[n]})$.

2.6 Multi-key homomorphic encryption with distributed setup

Here, we recall the definitions for multi-key HE (MKHE) with distributed setup. In this paper, we will treat both MKHE schemes for circuits and for branching programs. Below, we only give the formal definitions for the former since recovering the definitions for the latter is straightforward given those for the former.

In this paper, we will also make use of ordinary single-key HE (without setup). We review the formal definition of this type of HE in Sect. 2.3.

Definition 2.6 *(Multi-key HE with Distributed Setup, adapted from* [15, 17]*)* An MKHE scheme with distributed setup MKHE for a class of circuits $\mathcal{C}$, consists of the five algorithms $\mathsf{MKHE} = (\mathsf{dSetup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ with the following properties:

- (Syntax)
  - $\mathsf{pp}_i \xleftarrow{\$} \mathsf{dSetup}(1^\lambda, 1^N, i)$: This is the distributed setup algorithm that takes a security parameter 1^λ, the maximal number of inputs N (in unary), and an index $i \in [N]$ as

input, and outputs the i-th user's public parameter pp_i. We require that given $i \in [N]$ and pp, whether $\mathsf{pp} \in [\mathsf{dSetup}(1^\lambda, 1^N, i)]$ or not is efficiently checkable. [5]

- $(\mathsf{pk}_i, \mathsf{sk}_i) \xleftarrow{\$} \mathsf{KG}((\mathsf{pp}_j)_{j\in[N]}, i)$: This is the key generation algorithm that takes N public parameters $(\mathsf{pp}_i)_{i\in[N]}$ and an index $i \in [N]$ as input, and outputs the i-th user's public/secret key pair $(\mathsf{pk}_i, \mathsf{sk}_i)$.
- $\mathsf{ct} \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}_i, x)$: This is the encryption algorithm that takes a public key pk_i and a plaintext $x \in \{0,1\}$ as input, and outputs a ciphertext ct.
- $\widehat{\mathsf{ct}} \xleftarrow{\$} \mathsf{Eval}(C, (\mathsf{pp}_j)_{j\in[N]}, (\mathsf{pk}_j)_{j\in[N]}, \{(I_k, \mathsf{ct}_k)\}_{k\in[n]})$: This is the homomorphic evaluation algorithm that takes as input a circuit $C \in \mathcal{C}$ (with domain $\{0,1\}^n$), N public parameters $(\mathsf{pp}_j)_{j\in[N]}$, N public keys $(\mathsf{pk}_i)_{i\in[N]}$, and n pairs $(I_k, c_k)_{k\in[n]}$, where $I_k \in [N]$ and ct_k is presumed to be a ciphertext under pk_{I_k}. Then, this algorithm outputs an evaluated ciphertext $\widehat{\mathsf{ct}}$.
- $\widehat{x} := \mathsf{Dec}((\mathsf{sk}_i)_{i\in[N]}, \widehat{\mathsf{ct}})$: This is the (deterministic) decryption algorithm that takes N secret keys $(\mathsf{sk}_i)_{i\in[N]}$ and an evaluated ciphertext [6] $\widehat{\mathsf{ct}}$ as input, and outputs a decryption result $\widehat{x}$.

- (Correctness) For any $N \in \mathbb{N}$, any circuit $C \in \mathcal{C}$ (with n-bit input), any indices $I_1, \ldots, I_n \in [N]$, and any bits $x_1, \ldots, x_n \in \{0,1\}$, the following holds: If $\mathsf{pp}_j \xleftarrow{\$} \mathsf{dSetup}(1^\lambda, 1^N, j)$ for all $j \in [N]$, $(\mathsf{pk}_j, \mathsf{sk}_j) \xleftarrow{\$} \mathsf{KG}((\mathsf{pp}_j)_{j\in[N]}, j)$ for all $j \in [N]$, $\mathsf{ct}_k \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}_{I_k}, x_k)$ for all $k \in [n]$, and $\widehat{\mathsf{ct}} \xleftarrow{\$} \mathsf{Eval}(C, (\mathsf{pp}_j)_{j\in[N]}, (\mathsf{pk}_j)_{j\in[N]}, (I_k, \mathsf{ct}_k)_{k\in[n]})$, then we have

$$\Pr[\mathsf{Dec}((\mathsf{sk}_j)_{j\in[N]}, \widehat{\mathsf{ct}}) \neq C(x_1, \ldots, x_n)] = \mathsf{negl}(\lambda).$$

- (Semantic Security) For any PPT adversary $\mathcal{A}$, we have

$$|\Pr[\mathsf{Exp}_{\mathsf{MKHE},\mathcal{A}}(\lambda, 0) \to 1] - \Pr[\mathsf{Exp}_{\mathsf{MKHE},\mathcal{A}}(\lambda, 1) \to 1]| = \mathsf{negl}(\lambda),$$

where $\mathsf{Exp}_{\mathsf{MKHE},\mathcal{A}}(\lambda, b)$ for $b \in \{0,1\}$ is the following experiment run between the challenger and the (rushing) adversary $\mathcal{A}$ (namely, $\mathcal{A}$ generates corrupted public parameters after seeing an honest public parameter):

1. $\mathcal{A}$ chooses the maximal number of inputs N and the challenge index $i \in [N]$, and sends them to the challenger.
2. The challenger returns public parameters $\mathsf{pp}_i \xleftarrow{\$} \mathsf{dSetup}(1^\lambda, 1^N, i)$.
3. $\mathcal{A}$ chooses public parameters pp_j for all $j \in [N]\backslash\{i\}$, and sends $\{\mathsf{pp}_j\}_{j\in[N]\backslash\{i\}}$ to the challenger.
4. The challenger generates $(\mathsf{pk}_i, \mathsf{sk}_i) \xleftarrow{\$} \mathsf{KG}((\mathsf{pp}_j)_{j\in[N]}, i)$ and sends pk_i to $\mathcal{A}$.
5. $\mathcal{A}$ sends the challenge plaintexts x_0, x_1 to the challenger.
6. The challenger returns the challenge ciphertext $\mathsf{ct}^* \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}_i, x_b)$ to $\mathcal{A}$.
7. $\mathcal{A}$ outputs its guess $b' \in \{0,1\}$, which is treated as output of the experiment $\mathsf{Exp}_{\mathsf{MKHE},\mathcal{A}}(\lambda, b)$.

We say that MKHE is *weakly circular secure* if it remains secure even if bit-wise encryptions of the secret key sk_i are attached to the public key pk_i. We say that MKHE is *fully homomorphic*

[5] This property is trivially satisfied by the MKFHE scheme with distributed setup by Brakerski et al. [15].

[6] For simplicity, in this paper we only consider the decryption algorithm for evaluated ciphertexts for (single-key/multi-key) HE schemes.

(and call it *multi-key fully homomorphic encryption (MKFHE)*) if $\mathcal{C}$ is a set of all circuits of polynomial size. We say that MKHE is *compact* if there exists a polynomial p such that $|\widehat{\mathsf{ct}}| < p(\lambda, N)$, which is independent of C and n.

To ease notation, for an MKHE scheme with distributed setup, we will often write **pp**, **pk**, and **sk** to mean N public parameters $(\mathsf{pp}_j)_{j\in[N]}$, N public keys $(\mathsf{pk}_j)_{j\in[N]}$, and N secret keys $(\mathsf{sk}_j)_{j\in[N]}$, respectively. ($N$ will always be clear from the context.)

Bootstrapping Gentry's bootstrapping theorem [22] provides a way to extend the class of functions a HE scheme supports from a limited class to an unlimited class. This involves homomorphic evaluation of so-called the *augmented decryption function*. Here we recall its multi-key variant.

Definition 2.7 *(Multi-key Augmented Decryption Function)* Let MKHE = (dSetup, KG, Enc, Eval, Dec) be an MKHE scheme (for some circuit class). For hardwired values $\widehat{\mathsf{ct}}$, $\widehat{\mathsf{ct}}'$, the multi-key augmented decryption function $h_{\widehat{\mathsf{ct}},\widehat{\mathsf{ct}}'}$ is defined by

$$h_{\widehat{\mathsf{ct}},\widehat{\mathsf{ct}}'}((\mathsf{sk}_j)_{j\in[N]}) := \overline{\mathsf{Dec}((\mathsf{sk}_j)_{j\in[N]}, \widehat{\mathsf{ct}}) \wedge \mathsf{Dec}((\mathsf{sk}_j)_{j\in[N]}, \widehat{\mathsf{ct}}')}.$$

That is, the function $h_{\widehat{\mathsf{ct}},\widehat{\mathsf{ct}}'}$ interprets its input as N secret keys $(\mathsf{sk}_j)_{j\in[N]}$, decrypts the hardwired ciphertexts $\widehat{\mathsf{ct}}$ and $\widehat{\mathsf{ct}}'$, and returns the NAND of the results.

López-Alt et al. [31] showed a multi-key analogue of the bootstrapping theorem, which states that if an MKHE scheme supports homomorphic evaluation of circuits that can compute the multi-key augmented decryption function, and is furthermore (weakly) circular secure, then the scheme can be turned into a fully homomorphic one.

3 Privately expandable MKHE with distributed setup

In this section, we present an LWE-based MKFHE scheme with distributed setup, which additionally satisfies the privacy notion of *private expandability*. Specifically, we firstly extend the definition of private expandability for MKHE by [17] to schemes with a distributed setup, and then show that the MKFHE scheme with distributed setup by [15] can be modified to satisfy this notion, by introducing an alternative ciphertext expansion algorithm based on the approach of [17]. This modified scheme will be the main building block for our constructions of maliciously circuit-private MKHE and MKFHE presented in Sect. 4.

This section is organized as follows. In Sect. 3.1, we first formalize private expandability for MKHE with distributed setup. Next, in Sect. 3.2, we recall the MKFHE scheme with distributed setup by Brakerski et al. [15]. Then, in Sect. 3.3, we show how to make it privately expandable. Throughout this section, let $N = N(\lambda) \in \mathbb{N}$ be a polynomial denoting the number of users.

3.1 Private expandability of MKHE with distributed setup

Here, we define private expandability of MKHE with distributed setup, extending the original definition by [17]. For ease of exposition, we firstly introduce plain (possibly non-private) expandability, and then proceed to define private expandability.

Definition 3.1 *(Expandability)* A MKHE scheme with distributed setup (dSetup, KG, Enc, Eval, Dec) for a circuit class $\mathcal{C}$ is *expandable* if there exist two algorithms (Expand, $\widetilde{\mathsf{Eval}}$) with the following properties:

- (Syntax)
  - $\widetilde{\mathsf{ct}} \xleftarrow{\$} \mathsf{Expand}(\mathbf{pp}, \mathbf{pk}, i, \mathsf{ct})$: This is the expansion algorithm that takes N public parameters $\mathbf{pp}$, N public keys $\mathbf{pk}$, an index $i \in [N]$, and a ciphertext ct as input, and outputs an expanded ciphertext $\widetilde{\mathsf{ct}}$.
  - $\widehat{\mathsf{ct}} \xleftarrow{\$} \widetilde{\mathsf{Eval}}(C, \mathbf{pp}, \mathbf{pk}, (\widetilde{\mathsf{ct}}_k)_{k\in[n]})$: This is the homomorphic evaluation algorithm for expanded ciphertexts. It takes a circuit $C \in \mathcal{C}$ (with n-bit input), N public parameters $\mathbf{pp}$, N public keys $\mathbf{pk}$, and n expanded ciphertexts $(\widetilde{\mathsf{ct}}_k)_{k\in[n]}$ as input [7], and outputs an evaluated ciphertext $\widehat{\mathsf{ct}}$.
- (Correctness) For any circuit $C \in \mathcal{C}$ (with n-bit input), any plaintexts $x_1, \ldots, x_n \in \{0, 1\}$, and indices $I_1, \ldots, I_n \in [N]$, if $\mathsf{pp}_j \xleftarrow{\$} \mathsf{dSetup}(1^\lambda, 1^N, j)$ for all $j \in [N]$, $(\mathsf{pk}_j, \mathsf{sk}_j) \xleftarrow{\$} \mathsf{KG}(\mathbf{pp}, j)$ for all $j \in [N]$, $\mathsf{ct}_k \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}_{I_k}, x_k)$ for all $k \in [n]$, $\widetilde{\mathsf{ct}}_k \xleftarrow{\$} \mathsf{Expand}(\mathbf{pp}, \mathbf{pk}, I_k, \mathsf{ct}_k)$ for all $k \in [n]$, and $\widehat{\mathsf{ct}} \xleftarrow{\$} \mathsf{Eval}(\mathbf{pp}, \mathbf{pk}, (\widetilde{\mathsf{ct}}_k)_{k\in[n]})$, then

$$\Pr\left[\begin{array}{c}\exists k \in [n] : \mathsf{Dec}(\mathbf{sk}, \widetilde{\mathsf{ct}}_k) \neq x_k \\ \vee\ \mathsf{Dec}(\mathbf{sk}, \widehat{\mathsf{ct}}) \neq C(x_1, \ldots, x_n)\end{array}\right] = \mathsf{negl}(\lambda).$$

As in [17], for an expandable MKHE scheme with distributed setup, we overload the notation for the algorithms, and replace the evaluation algorithm Eval of the original scheme with $\widetilde{\mathsf{Eval}}$. Hence, when we consider a MKHE scheme to consist of the six algorithms $(\mathsf{dSetup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Expand}, \mathsf{Eval}, \mathsf{Dec})$, $\widetilde{\mathsf{Eval}}$ is the homomorphic evaluation algorithm for expanded ciphertexts (corresponding to $\widetilde{\mathsf{Eval}}$ above).

Definition 3.2 *(Private Expandability)* Let $\mathsf{MKHE} = (\mathsf{dSetup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Expand}, \mathsf{Eval}, \mathsf{Dec})$ be an expandable MHFE scheme with distributed setup. We say that MKHE is *privately expandable* if it satisfies the following property: For all $j \in [N]$, let $\mathsf{pp}_j \in [\mathsf{dSetup}(1^\lambda, 1^N, j)]$ and $(\mathsf{pk}_j, \mathsf{sk}_j) \in [\mathsf{KG}(\mathbf{pp}, j)]$.

Let $i, i' \in [N]$, $x \in \{0, 1\}$, $\mathsf{ct}_i \in [\mathsf{Enc}(\mathsf{pk}_i, x)]$, and $\mathsf{ct}_{i'} \in [\mathsf{Enc}(\mathsf{pk}_{i'}, x)]$. Then, we have

$$\mathsf{Expand}(\mathbf{pp}, \mathbf{pk}, i, \mathsf{ct}_i) \approx_s \mathsf{Expand}(\mathbf{pp}, \mathbf{pk}, i', \mathsf{ct}_{i'}),$$

where the statistical indistinguishability is guaranteed only by the random coins of Expand.

3.2 LWE-based expandable MKFHE by brakerski et al.

We now present the MKFHE scheme of [15] which we will denote $\mathsf{MKHE}_{\mathsf{BHP}}$. Although the original scheme has expandability, how this is supported is only informally sketched in [15]. Below we give a full description of the scheme which explicitly describes the expansion algorithm. The expandable MKFHE scheme $\mathsf{MKHE}_{\mathsf{BHP}} = (\mathsf{dSetup}_{\mathsf{BHP}}, \mathsf{KG}_{\mathsf{BHP}}, \mathsf{Enc}_{\mathsf{BHP}}, \mathsf{Expand}_{\mathsf{BHP}}, \mathsf{Eval}_{\mathsf{BHP}}, \mathsf{Dec}_{\mathsf{BHP}})$ is as follows.

- $\mathsf{dSetup}_{\mathsf{BHP}}(1^\lambda, 1^N, i \in [N])$: Let $n = n(\lambda)$ be a polynomial in security parameter λ, $q = q(\lambda)$ be a superpolynomial in λ, $\ell := \lceil \log q \rceil$, $m = O(n\ell)$, $w := m\ell$, and D be a Gaussian distribution whose samples have norm bounded by some polynomial $B = B(\lambda) \in \mathbb{N}$ except with negligible probability. Sample a matrix $\mathbf{A}_i \xleftarrow{\$} \mathbb{Z}_q^{m\times n}$, and output $\mathsf{pp}_i := \mathbf{A}_i$. (Note that any member in $\mathbb{Z}_q^{m\times n}$ is a possible public parameter.)

[7] Note that $\widetilde{\mathsf{Eval}}$ does not take indices I_k for expanded ciphertexts $\widetilde{\mathsf{ct}}_k$ as input.

- $\mathsf{KG}_{\mathsf{BHP}}(\mathbf{pp}, i \in [N])$: Sample $\mathbf{s}_i \xleftarrow{\$} \{0,1\}^{m-1}$, and set $\mathbf{t}_i^T := [\mathbf{s}_i^T \| 1] \in \mathbb{Z}_q^{1\times m}$. For every $j \in [N]$, compute $\mathbf{b}_{i,j}^T := \mathbf{t}_i^T \mathbf{A}_j \in \mathbb{Z}_q^{1\times n}$ and

$$\mathbf{B}_{i,j} := \mathbf{A}_i - \begin{bmatrix} \mathbf{0}_{(m-1)\times n} \\ \mathbf{b}_{i,j}^T \end{bmatrix} \in \mathbb{Z}_q^{m\times n},$$

which satisfies $\mathbf{t}_i^T \mathbf{B}_{i,j} = \mathbf{b}_{i,i}^T - \mathbf{b}_{i,j}^T \in \mathbb{Z}_q^{1\times n}$. Let $\mathbf{B}_i := \mathbf{B}_{i,i}$.

For every $k \in [m]$, sample $\mathbf{R}_k \xleftarrow{\$} D^{n\times w}$ and $\mathbf{E}_k \xleftarrow{\$} D^{m\times w}$, and compute an encryption of the k-th bit $\mathbf{t}_i[k]$ of the secret key $\mathbf{t}_i$ by

$$\mathbf{T}_{i,k} := \mathbf{B}_i \mathbf{R}_k + \mathbf{E}_k + \mathbf{t}_i[k]\mathbf{G} \in \mathbb{Z}_q^{m\times w}.$$

Output $\mathsf{pk}_i := ((\mathbf{b}_{i,j})_{j\in[N]}, (\mathbf{T}_{i,k})_{k\in[m]})$ and $\mathsf{sk}_i := \mathbf{t}_i$.

- $\mathsf{Enc}_{\mathsf{BHP}}(\mathsf{pk}_i, x \in \{0,1\})$: Sample $\mathbf{R} \xleftarrow{\$} D^{n\times w}$ and $\mathbf{E} \xleftarrow{\$} D^{m\times w}$, and compute

$$\mathbf{C} := \mathbf{B}_i \mathbf{R} + \mathbf{E} + x\mathbf{G} \in \mathbb{Z}_q^{m\times w}.$$

For every $\tau \in [n]$ and $k \in [w]$, sample $\mathbf{R}_{\tau,k} \xleftarrow{\$} D^{n\times w}$ and $\mathbf{E}_{\tau,k} \xleftarrow{\$} D^{m\times w}$, and compute an encryption of the (τ, k)-entry $\mathbf{R}[\tau, k]$ of $\mathbf{R}$ by

$$\mathbf{U}_{\tau,k} := \mathbf{B}_i \mathbf{R}_{\tau,k} + \mathbf{E}_{\tau,k} + \mathbf{R}[\tau,k]\mathbf{G} \in \mathbb{Z}_q^{m\times w}.$$

Output $\mathsf{ct}_i := (\mathbf{C}, (\mathbf{U}_{\tau,k})_{\tau\in[n],k\in[w]})$.

- $\mathsf{Expand}_{\mathsf{BHP}}(\mathbf{pp}, \mathbf{pk}, i, \mathsf{ct}_i)$: This algorithm uses the "linear combination" algorithm LComb described below as a sub-algorithm. Roughly, LComb takes encryptions of the randomness matrix, i.e. $(\mathbf{U}_{\tau,k})_{\tau\in[n],k\in[w]}$, used to encrypt a message, and some input vector in $\mathbb{Z}_q^n$ as input, and generates an encryption of the multiplication between the randomness matrix and the input vector. This is used to erase junk terms in decryption.
  - $\mathsf{LComb}((\mathbf{U}_{\tau,k} \in \mathbb{Z}_q^{m\times w})_{\tau\in[n],k\in[w]}, \mathbf{v} \in \mathbb{Z}_q^n)$: First define a matrix $\mathbf{Z}_{\tau,k}^{(\mathbf{v})} \in \mathbb{Z}^{n\times w}$ for $\tau \in [n]$ and $k \in [w]$ as

$$\mathbf{Z}_{\tau,k}^{(\mathbf{v})}[a,b] := \begin{cases} v_\tau \text{ if } a = n \text{ and } b = k \\ 0 \text{ otherwise,} \end{cases}$$

then output $\mathbf{X} := \sum_{\tau\in[n],k\in[w]} \mathbf{U}_{\tau,k}\mathbf{G}^{-1}(\mathbf{Z}_{\tau,k}^{(\mathbf{v})}) \in \mathbb{Z}_q^{m\times w}$.

Using LComb, $\mathsf{Expand}_{\mathsf{BHP}}$ proceeds as follows: For every $j \in [N] \setminus \{i\}$, compute

$$\mathbf{X}_j := \mathsf{LComb}((\mathbf{U}_{\tau,k})_{\tau\in[n],k\in[w]}, \mathbf{b}_{i,i} - \mathbf{b}_{j,i}).$$

Then, output the expanded ciphertext $\widetilde{\mathbf{C}} \in \mathbb{Z}_q^{N\cdot m\times N\cdot w}$ that consists of N^2 submatrices of size $m \times w$, has the input ciphertext $\mathbf{C}$ in diagonal, and $\mathbf{X}_j$ in the (i, j)-th submatrix for $j \in [N] \setminus \{i\}$. More visually, the expanded ciphertext $\widetilde{\mathbf{C}}$ is of the following form

$$\widetilde{\mathbf{C}} := \begin{bmatrix} \mathbf{C} & & & & & & \\ & \ddots & & & & & \\ & & \mathbf{C} & & & & \\ \mathbf{X}_1 & \cdots & \mathbf{X}_{i-1} & \mathbf{C} & \mathbf{X}_{i+1} & \cdots & \mathbf{X}_N \\ & & & \mathbf{C} & & & \\ & & & & \ddots & & \\ & & & & & & \mathbf{C} \end{bmatrix} \in \mathbb{Z}_q^{N\cdot m\times N\cdot w},$$

where all the empty submatrices are zero matrices.

- $\mathsf{Eval}_{\mathsf{BHP}}(C, \mathbf{pp}, \mathbf{pk}, (\widetilde{\mathbf{C}}_k)_{k\in[n]})$: This algorithm homomorphically evaluates the given circuit C over input ciphertexts $(\widetilde{\mathbf{C}}_k)_{k\in[n]}$ gate-by-gate (assuming C is described via NAND gates only), and outputs the final result as an evaluated ciphertext $\widehat{\mathsf{ct}}$. To implement the NAND evaluation algorithm NAND, the algorithm makes use of homomorphic addition Add and multiplication Mult algorithms described below.
  - $\mathsf{Add}(\widetilde{\mathbf{C}}_1, \widetilde{\mathbf{C}}_2)$: Homomorphic addition can be just computed by adding the expanded ciphertext matrices: $\widetilde{\mathbf{C}}_{\mathsf{Add}} := \widetilde{\mathbf{C}}_1 + \widetilde{\mathbf{C}}_2 \in \mathbb{Z}_q^{N\cdot m\times N\cdot w}$. We can see correctness of this homomorphic addition from the condition satisfied for the expanded ciphertexts.
  - $\mathsf{Mult}(\widetilde{\mathbf{C}}_1, \widetilde{\mathbf{C}}_2)$: Homomorphic multiplication is computed by first decomposing the expanded ciphertexts by the gadget inverse function and multiplying it to the normal expanded ciphertext: $\widetilde{\mathbf{C}}_{\mathsf{Mult}} := \widetilde{\mathbf{C}}_1\mathbf{G}^{-1}(\widetilde{\mathbf{C}}_2) \in \mathbb{Z}_q^{N\cdot m\times N\cdot w}$. Also, correctness of the homomorphic multiplication can be easily checked from the relation between expanded ciphertexts and the concatenation of the involved secret keys.
  - $\mathsf{NAND}(\mathbf{pp}, \mathbf{pk}, \widetilde{\mathbf{C}}_1, \widetilde{\mathbf{C}}_2)$: For all $j \in [N], k \in [m]$, compute the expanded bootstrapping keys $\widetilde{\mathbf{T}}_{j,k} := \mathsf{Expand}_{\mathsf{BHP}}(\mathbf{pp}, \mathbf{pk}, j, \mathbf{T}_{j,k})$. [8] Compute a ciphertext $\widetilde{\mathbf{C}}_{\mathsf{NAND}}$ by evaluating the multi-key augmented decryption function $h_{\widetilde{\mathbf{C}}_1,\widetilde{\mathbf{C}}_2}$ (in Definition 2.7) on $\widetilde{\mathbf{T}}_{j,k}$'s gate-by-gate via Add and Mult.
- $\mathsf{Dec}_{\mathsf{BHP}}(\mathbf{sk}, \widehat{\mathsf{ct}})$: Parse $\widehat{\mathsf{ct}}$ as $\widetilde{\mathbf{C}} \in \mathbb{Z}_q^{N\cdot m\times N\cdot w}$.
  Set $\widetilde{\mathbf{t}}^T := [\mathbf{t}_1^T \| \cdots \| \mathbf{t}_N^T]$, and output $x' := \lfloor (2/q) \cdot \widetilde{\mathbf{t}}^T \widetilde{\mathbf{C}} \mathbf{u}_{N\cdot w} \rceil$, where $\mathbf{u}_{N\cdot w} := (0, \ldots, 0, 1) \in \{0,1\}^{N\cdot w}$.

The above completes the description of $\mathsf{MKHE}_{\mathsf{BHP}}$. For completeness, we show the correctness of the above described expansion algorithm $\mathsf{Expand}_{\mathsf{BHP}}$, namely, that an expanded ciphertext output from $\mathsf{Expand}_{\mathsf{BHP}}$ can be correctly decrypted.

Lemma 3.1 (Correctness of $Expand_{BHP}$) *Let $i \in [N]$ and $x \in \{0,1\}$. If $pp_j \xleftarrow{\$} dSetup_{BHP}(1^\lambda, 1^N, j)$ for every $j \in [N]$, $(pk_j, sk_j) \xleftarrow{\$} KG_{BHP}(\mathbf{pp}, j)$ for every $j \in [N]$, $ct_i \xleftarrow{\$} Enc_{BHP}(pk_i, x)$, and $\widetilde{\mathbf{C}}_i := Expand_{BHP}(\mathbf{pp}, \mathbf{pk}, i, ct_i)$, then*

$$\Pr\left[Dec_{BHP}(\mathbf{sk}, \widetilde{\mathbf{C}}_i) \neq x\right] = negl(\lambda).$$

We first give the following property of LComb. This is an analogy of [16, Lemma A.3], showing an analogous property of the linear combination algorithm for [35].

The proof is almost the same as the one of [16] except the estimation of noise growth incurred by this algorithm.

Claim 3.1 *Let $\mathbf{R} \in \mathbb{Z}^{n\times w}$ be a matrix, and $\mathbf{t}^T = [\mathbf{s}^T, 1] \in \{0,1\}^m$ for $\mathbf{s} \in \{0,1\}^{m-1}$. For $\tau \in [n]$ and $k \in [w]$, let $\mathbf{U}_{\tau,k} \in \mathbb{Z}_q^{m\times w}$ be such that $\mathbf{t}^T\mathbf{U}_{\tau,k} = \mathbf{e}_{\tau,k}^T + \mathbf{t}^T\mathbf{R}[\tau,k]\mathbf{G}$ for $\mathbf{R}[\tau,k] \in \mathbb{Z}_B$ and $\mathbf{e}_{\tau,k} \in \mathbb{Z}^w$ with $\|\mathbf{e}_{\tau,k}\| < mB$. Let $\mathbf{b} = (b_1, \ldots, b_n) \in \mathbb{Z}_q^n$ and $\mathbf{X} = LComb((\mathbf{U}_{\tau,k})_{\tau,k}, \mathbf{b})$. Then we have*

$$\mathbf{t}^T\mathbf{X} = \mathbf{e}''^T + \mathbf{b}^T\mathbf{R} \in \mathbb{Z}_q^w$$

for some $\mathbf{e}''^T \in \mathbb{Z}^w$ with $\|\mathbf{e}''^T\| < nw^2mB$.

[8] The expanded bootstrapping keys need to be computed only once and are reused across multiple calls of NAND.

Proof of Claim For the above vector $\mathbf{t}$ and matrix $\mathbf{X}$, it holds that

$$\begin{aligned}\mathbf{t}^T\mathbf{X} &= \mathbf{t}^T \sum_{\tau\in[n],k\in[w]} \mathbf{U}_{\tau,k}\mathbf{G}^{-1}(\mathbf{Z}^{(\mathbf{b})}_{\tau,k}) \\ &= \sum_{\tau,k}(\mathbf{e}^T_{\tau,k} + \mathbf{t}^T\mathbf{R}[\tau,k]\mathbf{G})\mathbf{G}^{-1}(\mathbf{Z}^{(\mathbf{b})}_{\tau,k}) \\ &= \sum_{\tau,k}(\mathbf{e}'^T_{\tau,k} + \mathbf{t}^T\mathbf{R}[\tau,k]\mathbf{Z}^{(\mathbf{b})}_{\tau,k}) \\ &= \mathbf{e}''^T + \mathbf{t}^T\sum_{\tau,k}\mathbf{R}[\tau,k]\mathbf{Z}^{(\mathbf{b})}_{\tau,k},\end{aligned}$$

where $\mathbf{e}''^T = \sum_{\tau\in[n],k\in[w]}\mathbf{e}'^T_{\tau,k}$ satisfying $\|\mathbf{e}''\| < nw^2mB$, and

$$\sum_{\tau,k}\mathbf{R}[\tau,k]\mathbf{Z}^{(\mathbf{b})}_{\tau,k} = \begin{bmatrix} 0 & \cdots & 0 \\ \vdots & & \vdots \\ 0 & \cdots & 0 \\ \sum_\tau \mathbf{R}[\tau,1]\cdot b_\tau & \cdots & \sum_\tau \mathbf{R}[\tau,w]\cdot b_\tau \end{bmatrix} = \begin{bmatrix}\mathbf{0} \\ \mathbf{b}^T\mathbf{R}\end{bmatrix}.$$

Therefore we have

$$\mathbf{t}^T\mathbf{X} = \mathbf{e}''^T + [\mathbf{s}^T, 1]\begin{bmatrix}\mathbf{0} \\ \mathbf{b}^T\mathbf{R}\end{bmatrix} = \mathbf{e}''^T + \mathbf{b}^T\mathbf{R}.$$

□

We now turn to the proof of the correctness of $\mathsf{Expand}_{\mathsf{BHP}}$. Let $\widetilde{\mathbf{t}}^T := [\mathbf{t}_1^T\|\cdots\|\mathbf{t}_N^T] \in \{0,1\}^{N\cdot m}$ be the concatenation of the secret keys, $\mathsf{ct}_i = (\mathbf{C}, (\mathbf{U}_{j,k})_{j\in[n],k\in[w]})$ be output by $\mathsf{Enc}_{\mathsf{BHP}}(\mathsf{pk}_i, x)$, and $\widetilde{\mathbf{C}} := \mathsf{Expand}_{\mathsf{BHP}}(\mathbf{pp}, \mathbf{pk}, i, \mathsf{ct}_i)$. Then

$$\begin{aligned}\widetilde{\mathbf{t}}^T\widetilde{\mathbf{C}} &= \left[\mathbf{t}_1^T\|\cdots\|\mathbf{t}_N^T\right]\cdot\begin{bmatrix}\mathbf{C} & & & & & & \\ & \ddots & & & & & \\ & & \mathbf{C} & & & & \\ \mathbf{X}_1 & \cdots & \mathbf{X}_{i-1} & \mathbf{C} & \mathbf{X}_{i+1} & \cdots & \mathbf{X}_N \\ & & & & \mathbf{C} & & \\ & & & & & \ddots & \\ & & & & & & \mathbf{C}\end{bmatrix} \\ &= \left[\mathbf{t}_1^T\mathbf{C}+\mathbf{t}_i^T\mathbf{X}_1\|\cdots\|\mathbf{t}_{i-1}^T\mathbf{C}+\mathbf{t}_i^T\mathbf{X}_{i-1}\|\mathbf{t}_i^T\mathbf{C}\|\mathbf{t}_{i+1}^T\mathbf{C}+\mathbf{t}_i^T\mathbf{X}_{i+1}\|\cdots\|\mathbf{t}_N^T\mathbf{C}+\mathbf{t}_i^T\mathbf{X}_N\right]\end{aligned}$$

where $\mathbf{X}_j = \mathsf{LComb}((\mathbf{U}_{j,k})_{j\in[n],k\in[w]}, \mathbf{b}_{i,i} - \mathbf{b}_{j,i})$ for each $j \in [N]\setminus\{i\}$. By the above claim, for every $j \in [N]$, we have $\mathbf{t}_j^T\mathbf{C} = (\mathbf{b}_{j,i} - \mathbf{b}_{i,i})^T\mathbf{R} + \mathbf{t}_j^T\mathbf{E}_i + x\cdot\mathbf{t}_j^T\mathbf{G}$, and $\mathbf{t}_i^T\mathbf{X}_j = \mathbf{e}_j^T + (\mathbf{b}_{i,i} - \mathbf{b}_{j,i})^T\mathbf{R}$ for some matrix $\mathbf{R}$ (randomness of Enc), and noise $\mathbf{e}_j$ and $\mathbf{E}_i$ such that $\|\mathbf{e}_j\| < nw^2mB$ and $\|\mathbf{E}_i\| < B$ with high probability. Thus, it holds that

$$\widetilde{\mathbf{t}}^T\widetilde{\mathbf{C}} = \left[\mathbf{t}_1^T\mathbf{E}_i+\mathbf{e}_1^T\|\cdots\|\mathbf{t}_{i-1}^T\mathbf{E}_i+\mathbf{e}_{i-1}^T\|\mathbf{t}_i^T\mathbf{E}_i\|\mathbf{t}_{i+1}^T\mathbf{E}_i+\mathbf{e}_{i+1}^T\|\cdots\|\mathbf{t}_N^T\mathbf{E}_i+\mathbf{e}_N^T\right] + x\cdot\widetilde{\mathbf{t}}^T\mathbf{G},$$

which implies $\mathsf{Dec}_{\mathsf{BHP}}(\mathbf{sk}, \widetilde{\mathbf{C}}) = x$.

3.3 Making $\mathsf{MKHE}_{\mathsf{BHP}}$ privately expandable

It is straightforward to see that $\mathsf{MKHE}_{\mathsf{BHP}}$ is *not* privately expandable. Specifically, the positions of non-zero submatrices of an expanded ciphertext $\widetilde{\mathbf{C}}$ output from $\mathsf{Expand}_{\mathsf{BHP}}$ directly reveal the user index.

We now show a special ciphertext expansion algorithm $\mathsf{pExpand}_{\mathsf{BHP}}$ that can make $\mathsf{MKHE}_{\mathsf{BHP}}$ privately expandable. The simple idea behind the construction is to hide the zero submatrices in an expanded ciphertext by adding dummy ciphertexts of zero generated by using "larger" noise.

In addition to the algorithms $\mathsf{Expand}_{\mathsf{BHP}}$ and Add from the original scheme $\mathsf{MKHE}_{\mathsf{BHP}}$, $\mathsf{pExpand}_{\mathsf{BHP}}$ uses two sub-algorithms which we denote by Enc^* and Add^*. The former algorithm is the same as $\mathsf{Enc}_{\mathsf{BHP}}$ except that it uses "larger noise", and the latter is just the homomorphic addition algorithm over *pre-expanded* ciphertexts and is simply implemented by element-wise addition.

The formal description of the algorithms are as follows:

- $\mathsf{Enc}^*(\mathsf{pk}_i, x)$: Generate a ciphertext $\mathsf{ct} = (\mathbf{C}, (\mathbf{U}_{\tau,k})_{\tau\in[n],k\in[w]})$ in the same way as $\mathsf{Enc}_{\mathsf{BHP}}(\mathsf{pk}_i, x)$, except that each entry in $\mathbf{R}$, $\mathbf{E}$, $\mathbf{R}_{\tau,k}$, and $\mathbf{E}_{\tau,k}$ for every $\tau \in [n]$ and $k \in [w]$, is sampled from D_t where t is superpolynomial in λ.
- $\mathsf{Add}^*(\mathsf{ct}, \mathsf{ct}')$: Parse ct as $(\mathbf{C}, (\mathbf{U}_{\tau,k})_{\tau\in[n],k\in[w]})$ and ct' as $(\mathbf{C}', (\mathbf{U}'_{\tau,k})_{\tau\in[n],k\in[w]})$. Compute $\mathbf{C}'' := \mathbf{C} + \mathbf{C}' \in \mathbb{Z}_q^{m\times w}$ and $\mathbf{U}''_{\tau,k} := \mathbf{U}_{\tau,k} + \mathbf{U}'_{\tau,k} \in \mathbb{Z}_q^{m\times w}$ for all $\tau \in [n]$ and $k \in [w]$. Output $\mathsf{ct}'' := (\mathbf{C}'', (\mathbf{U}''_{\tau,k})_{\tau\in[n],k\in[w]})$.
- $\mathsf{pExpand}_{\mathsf{BHP}}(\mathbf{pp}, \mathbf{pk}, i, \mathsf{ct}_i)$: For every $j \in [N]$, compute

$$\mathsf{ct}^*_j := \begin{cases} \mathsf{Add}^*(\mathsf{ct}_i, \mathsf{Enc}^*(\mathsf{pk}_i, 0)) & \text{if } j = i \\ \mathsf{Enc}^*(\mathsf{pk}_j, 0) & \text{otherwise} \end{cases}.$$

  and $\widetilde{\mathsf{ct}}^*_j := \mathsf{Expand}_{\mathsf{BHP}}(\mathbf{pp}, \mathbf{pk}, j, \mathsf{ct}^*_j)$. Output $\widetilde{\mathsf{ct}}_i := \sum_{j\in[N]} \widetilde{\mathsf{ct}}^*_j$, where the summation of $\widetilde{\mathsf{ct}}^*_j$'s is realized by Add.

Since the algorithm Add^* just homomorphically adds fresh ciphertexts of $\mathsf{MKHE}_{\mathsf{BHP}}$ that corresponds to ciphertexts of the dual scheme of [23], we can easily check that correctness of Add^* holds. Hence, the correctness of $\mathsf{pExpand}_{\mathsf{BHP}}$ can be seen as follows.

Lemma 3.2 (Correctness of $pExpand_{BHP}$) *Let $i \in [N]$ and $x \in \{0,1\}$. If $pp_j \xleftarrow{\$} dSetup_{BHP}(1^\lambda, 1^N, j)$ for every $j \in [N]$, $(pk_j, sk_j) \xleftarrow{\$} KG_{BHP}(\mathbf{pp}, j)$ for every $j \in [N]$, $ct \xleftarrow{\$} Enc_{BHP}(pk_i, x)$, and $\widetilde{ct} := \widetilde{\mathbf{C}} \xleftarrow{\$} pExpand_{BHP}(\mathbf{pp}, \mathbf{pk}, i, ct)$, then*

$$\Pr\left[Dec_{BHP}(\mathbf{sk}, \widetilde{ct}) \neq x\right] = negl(\lambda).$$

Proof Let $\widetilde{\mathbf{t}}^T := [\mathbf{t}_1^T \| \cdots \| \mathbf{t}_N^T]$ where $\mathbf{t}_j = \mathsf{sk}_j$ for each $j \in [N]$. Then, by the correctness of Add and Add^*, we have $\widetilde{\mathbf{t}}^T\widetilde{\mathbf{C}} = (\sum_{j\in[N]} \widetilde{\mathbf{e}}'^T_j) + (\sum_{j\in[N]} x_j) \cdot \widetilde{\mathbf{t}}^T\mathbf{G}$, where $x_j = x$ if $j = i$, and $x_j = 0$ otherwise by the construction of $\mathsf{pExpand}_{\mathsf{BHP}}$. If $\|\widetilde{\mathbf{e}}\| < q/4$ for $\widetilde{\mathbf{e}} := \sum_{j\in[N]} \widetilde{\mathbf{e}}'_j$ (that can be satisfied by an appropriate choice of q), then $x = \mathsf{Dec}_{\mathsf{BHP}}(\mathbf{sk}, \widetilde{\mathsf{ct}})$. □

In the following, we show that the output of $\mathsf{pExpand}_{\mathsf{BHP}}$ hides user indices.

Lemma 3.3 (Privacy of $pExpand_{BHP}$) *For all $j \in [N]$, let $pp_j \in [dSetup_{BHP}(1^\lambda, 1^N, j)]$, and $(pk_j, sk_j) \in [KG_{BHP}(\mathbf{pp}, j)]$. Then, for any $i, i' \in [N]$, $x \in \{0,1\}$, $ct_i \in [Enc_{BHP}(pk_i, x)]$, and $ct_{i'} \in [Enc_{BHP}(pk_{i'}, x)]$, we have*

$$pExpand_{BHP}(\mathbf{pp}, \mathbf{pk}, i, ct_i) \approx_s pExpand_{BHP}(\mathbf{pp}, \mathbf{pk}, i', ct_{i'})$$

where the statistical indistinguishability is guaranteed only by the random coins of $\mathsf{pExpand}_{\mathsf{BHP}}$.

Proof We first show a simple fact about Add^* useful for showing the lemma.

Claim 3.2 *For all* $j \in [N]$, *let* $pp_j \in [dSetup_{BHP}(1^\lambda, 1^N, j)]$, *and* $(pk_j, sk_j) \in [KG_{BHP}(\mathbf{pp}, j)]$. *Then, for any* $i \in [N]$, $x \in \{0,1\}$, *and* $ct \in [Enc_{BHP}(pk_i, x)]$, *we have*

$$Add^*(ct, Enc^*(pk_i, 0)) \approx_s Enc^*(pk_i, x),$$

where the statistical indistinguishability is guaranteed only by the random coins of Enc^*.

Proof of the claim Let $r = (\mathbf{R}, \mathbf{E}, (\mathbf{R}_{\tau,k})_{\tau\in[n],k\in[w]}), (\mathbf{E}_{\tau,k})_{\tau\in[n],k\in[w]})$ be the randomness such that $\mathsf{ct} = \mathsf{Enc}_{\mathsf{BHP}}(\mathsf{pk}_i, x; r)$ with $\|\mathbf{R}\|, \|\mathbf{E}\|, \|\mathbf{R}_{\tau,k}\|, \|\mathbf{E}_{\tau,k}\| < r = O(\sqrt{n})$. Let D' be the distribution of a randomness in Enc^*, namely, every element is sampled from D_t. Then, by the designs of $\mathsf{Enc}_{\mathsf{BHP}}$ and Add^*, the distribution of the left hand side is identical to $\mathsf{Enc}^*(\mathsf{pk}_i, x; r+r')$ where $r' \xleftarrow{\$} D'$ and the addition of the randomnesses is just element-wise addition. By Lemma 2.1, the distribution $r + D' = \{r' \xleftarrow{\$} D' : r + r'\}$ (where r is fixed) is statistically close to D'. Hence, $\mathsf{Add}(\mathsf{ct}, \mathsf{Enc}^*(\mathsf{pk}_i, 0)) = \{r' \xleftarrow{\$} D' : \mathsf{Enc}^*(\mathsf{pk}_i, x; r+r')\} \approx_s \mathsf{Enc}^*(\mathsf{pk}_i, x)$. □

For $i'' \in \{i, i'\}$, let $\mathsf{pExpand}'(\mathbf{pp}, \mathbf{pk}, i'', x)$ be defined in the same way as $\mathsf{pExpand}_{\mathsf{BHP}}(\mathbf{pp}, \mathbf{pk}, i'', \mathsf{ct}_{i''})$ except that $\mathsf{ct}^*_{i''}$ is generated as $\mathsf{Enc}^*(\mathsf{pk}_{i''}, x)$ instead of $\mathsf{Add}^*(\mathsf{ct}_{i''}, \mathsf{Enc}^*(\mathsf{pk}_{i''}, 0))$. By the above claim, for both $i'' \in \{i, i'\}$, $\mathsf{ct}^*_{i''}$ generated in $\mathsf{pExpand}'$ and that in $\mathsf{pExpand}_{\mathsf{BHP}}$ are statistically indistinguishable, and thus we have $\mathsf{pExpand}_{\mathsf{BHP}}(\mathbf{pp}, \mathbf{pk}, i'', \mathsf{ct}_{i''}) \approx_s \mathsf{pExpand}'(\mathbf{pp}, \mathbf{pk}, i'', x)$. Hence, to prove the lemma, it is sufficient to show $\mathsf{pExpand}'(\mathbf{pp}, \mathbf{pk}, i, x) \approx_s \mathsf{pExpand}'(\mathbf{pp}, \mathbf{pk}, i', x)$. To this end, we argue similarly to the proof of [16, Lemma A.5] and show that the distribution of the output of $\mathsf{pExpand}'$ is identical.

For each $i'' \in \{i, i'\}$, let $\widetilde{\mathsf{ct}}_{i''}$ be an output of $\mathsf{pExpand}'(\mathbf{pp}, \mathbf{pk}, i'', x)$. If $i = i'$ then clearly $\widetilde{\mathsf{ct}}_i$ and $\widetilde{\mathsf{ct}}_{i'}$ are distributed identically. When $i \neq i'$, the only difference in the executions of $\mathsf{pExpand}'(\mathbf{pp}, \mathbf{pk}, i, x)$ and $\mathsf{pExpand}'(\mathbf{pp}, \mathbf{pk}, i', x)$ is in $\widetilde{\mathsf{ct}}^*_i$ and $\widetilde{\mathsf{ct}}^*_{i'}$ (other ciphertexts $\widetilde{\mathsf{ct}}^*_j$ for $j \in [N] \setminus \{i, i'\}$ are clearly distributed identically), and thus we only have to show that $\widetilde{\mathsf{ct}}^*_{i,i'} := \widetilde{\mathsf{ct}}^*_i + \widetilde{\mathsf{ct}}^*_{i'}$ generated in the two executions of $\mathsf{pExpand}'$ are distributed identically. $\mathsf{pExpand}'$ generates $\widetilde{\mathsf{ct}}^*_i$ of the form

$$\widetilde{\mathsf{ct}}^*_i = \begin{bmatrix} \mathbf{C}_i & & & & & & \\ & \ddots & & & & & \\ & & \mathbf{C}_i & & & & \\ \mathbf{X}_{i,1} & \cdots & \mathbf{X}_{i,i-1} & \mathbf{C}_i & \mathbf{X}_{i,i+1} & \cdots & \mathbf{X}_{i,N} \\ & & & \mathbf{C}_i & & & \\ & & & & \ddots & & \\ & & & & & & \mathbf{C}_i \end{bmatrix},$$

where $\mathbf{C}_i = \mathbf{B}_i\mathbf{R}_i + \mathbf{E}_i + x\mathbf{G}$ in the execution of $\mathsf{pExpand}'$ for i, and $\mathbf{C}_i = \mathbf{B}_i\mathbf{R}_i + \mathbf{E}_i$ in the execution for i'. For $i' \neq i$, $\widetilde{\mathsf{ct}}^*_{i'}$ has a similar form but $\mathbf{C}_{i'} = \mathbf{B}_{i'}\mathbf{R}_{i'} + \mathbf{E}_{i'}$ in the former execution, and $\mathbf{C}_{i'} = \mathbf{B}_{i'}\mathbf{R}_{i'} + \mathbf{E}_{i'} + x\mathbf{G}$ in the latter execution. $\mathbf{R}_i$ and $\mathbf{R}_{i'}$ are sampled from the same distribution and the same for $\mathbf{E}_i$ and $\mathbf{E}_{i'}$. Furthermore, in both executions, $\mathbf{X}_{i,j}$ for each $j \in [N] \setminus \{i\}$ is generated according to the same distribution, and the same is true for

$\mathbf{X}_{i',j}$ for each $j \in [N] \setminus \{i'\}$ Both executions generate $\widetilde{\mathsf{ct}}^*_{i,i'} = \widetilde{\mathsf{ct}}^*_i + \widetilde{\mathsf{ct}}^*_{i'}$ of the form (where w.l.o.g. we assume $i < i'$)

$$\widetilde{\mathsf{ct}}^*_{i,i'} = \begin{bmatrix} \mathbf{C}_{i,i'} & & & & & & \\ & \ddots & & & & & \\ \mathbf{X}_{i,1} & \cdots & \mathbf{C}_{i,i'} & \cdots & \mathbf{X}_{i,i'} & \cdots & \mathbf{X}_{i,N} \\ & & & \ddots & & & \\ \mathbf{X}_{i',1} & \cdots & \mathbf{X}_{i',i} & \cdots & \mathbf{C}_{i,i'} & \cdots & \mathbf{X}_{i',N} \\ & & & & & \ddots & \\ & & & & & & \mathbf{C}_{i,i'} \end{bmatrix},$$

where $\mathbf{C}_{i,i'} := \mathbf{B}_i\mathbf{R}_i + \mathbf{B}_{i'}\mathbf{R}_{i'} + (\mathbf{E}_i + \mathbf{E}_{i'}) + x\mathbf{G}$. Therefore, $\widetilde{\mathsf{ct}}^*_{i,i'}$ in the two executions of $\mathsf{pExpand}'$ are distributed identically. This means that the distribution of $\mathsf{pExpand}'(\mathbf{pp}, \mathbf{pk}, i, x)$ and that of $\mathsf{pExpand}'(\mathbf{pp}, \mathbf{pk}, i', x)$ are identical regardless of $i, i' \in [N]$. This in turn implies $\mathsf{pExpand}_{\mathsf{BHP}}(\mathbf{pp}, \mathbf{pk}, i, \mathsf{ct}_i) \approx_s \mathsf{pExpand}_{\mathsf{BHP}}(\mathbf{pp}, \mathbf{pk}, i', \mathsf{ct}_{i'})$. □

We summarize the consequence of the above lemmas.

Theorem 3.1 *Let* $\mathit{MKHE}_{\mathit{pBHP}} := (\mathit{dSetup}_{\mathit{BHP}}, \mathit{KG}_{\mathit{BHP}}, \mathit{Enc}_{\mathit{BHP}}, \mathit{pExpand}_{\mathit{BHP}}, \mathit{Eval}_{\mathit{BHP}}, \mathit{Dec}_{\mathit{BHP}})$ *be an expandable MKFHE scheme with distributed setup that is the same as* $\mathit{MKHE}_{\mathit{BHP}}$ *except that its expansion algorithm* $\mathit{Expand}_{\mathit{BHP}}$ *is replaced with* $\mathit{pExpand}_{\mathit{BHP}}$*. Then,* $\mathit{MKHE}_{\mathit{pBHP}}$ *is privately expandable.*

4 Maliciously circuit-private MKHE based on LWE

In this section, we show two constructions of maliciously circuit-private MKHE with distributed setup, one for branching programs, and the other for circuits of all poly-sized circuits (i.e. MKFHE), These constructions are based on the construction methodology of [17].

In Sect. 4.1, we give the formal definition of malicious circuit privacy for MKHE with distributed setup. In Sect. 4.2, we show the first construction: a maliciously circuit-private MKHE scheme with distributed setup for branching programs from the combination of a privately expandable MKFHE scheme with distributed setup and a maliciously circuit-private *single-key* FHE scheme. In Sect. 4.3, we show how to enhance maliciously circuit-private MKHE with distributed setup for branching program to fully homomorphic one by using a (non-circuit-private) MKFHE scheme as an additional building block. (There, we also explain a somewhat non-standard circular security assumption required to prove the semantic security of this MKFHE construction.)

Throughout this section, let $N = N(\lambda) \in \mathbb{N}$ be a polynomial denoting the number of users.

4.1 Definition

Here we introduce the definition of malicious circuit privacy for MKHE with distributed setup.

Definition 4.1 *(Malicious Circuit-privacy for MKHE with Distributed Setup, adapted from* [17]*)* Let $\mathsf{MKHE} = (\mathsf{dSetup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ be an MKHE scheme with distributed setup for a class of circuits $\mathcal{C}$. We say MKHE is *maliciously circuit-private* if there exists an

unbounded algorithm Sim (called the *simulator*) and an unbounded deterministic algorithm Ext (called the *extractor*) such that for all circuits $C \in \mathcal{C}$ (with n-bit input), all indices $I_1, \ldots, I_n \in [N]$, and all possibly malformed public parameters $\mathbf{pp}^* = (\mathsf{pp}_i^*)_{i\in[N]}$, public keys $\mathbf{pk}^* := (\mathsf{pk}_i^*)_{i\in[N]}$, and ciphertexts $\mathsf{ct}_1^*, \ldots, \mathsf{ct}_n^*$, we have

$$\mathsf{Eval}(C, \mathbf{pp}^*, \mathbf{pk}^*, (I_k, \mathsf{ct}_k^*)_{k\in[n]}) \approx_s \mathsf{Sim}(\mathbf{pp}^*, \mathbf{pk}^*, (I_k, \mathsf{ct}_k^*)_{k\in[n]}, C(x_1^*, \ldots, x_n^*)),$$

where $x_i^* := \mathsf{Ext}(\mathbf{pp}^*, I_k, \mathsf{pk}_{I_k}^*, \mathsf{ct}_k^*)$ for all $k \in [n]$.

4.2 Scheme for branching programs

We now show how to construct a maliciously circuit-private MKFHE scheme for branching programs. The construction is generic and based on the combination of a privately-expandable MKFHE scheme and a maliciously circuit-private single-key FHE scheme.

The main difference from [17] is that we need to treat public parameters $\mathbf{pp}$ generated by the distributed setup appropriately: We have an additional check in the homomorphic evaluation algorithm to make sure the public parameters pp are possible outputs of the distributed setup algorithm of the underlying privately-expandable MKFHE scheme. There is also a technically subtle but important difference in the design of the evaluation algorithm (see the explanation in the footnote of the scheme description).

Intuition of our construction In order to achieve malicious circuit-privacy for the homomorphic evaluation algorithm, we need to deal with (possibly) malformed input public keys and ciphertexts. To do this, we use a maliciously circuit-private single-key FHE scheme to homomorphically check whether the inputs are possible output of semi-honest execution of the key generation and encryption algorithm. This guarantees that the input public keys and ciphertexts for the homomorphic evaluation of a branching program are well-formed. Hence, in the following, we consider the semi-honest circuit-private evaluation of a branching program.

To explain the intuition of the circuit-private evaluation, first recall how to compute a branching program. Let $P = (G = (V, E), v_0, T, \phi_V, \phi_E)$ be a length-ℓ branching program over $\{0,1\}^n$. To evaluate P on $x \in \{0,1\}^n$, we follow the path induced by ϕ_V and x from the initial node v_0 to a terminal node in T. As defined in Definition 2.1, $P_v(x)$ refers to follow the path from any node $v \in V$, and $P_{v_0}(x) = P(x)$. Since every node $v \in V$ has two child nodes $v', v'' \in V$ such that $\phi_E(v, v') = 0$ and $\phi_E(v, v'') = 1$, $P_v(x)$ can be defined for $i = \phi_V(v)$ as $P_v(x) := P_{v'}(x)$ if $x_i = 0$, and $P_v(x) := P_{v''}(x)$ if $x_i = 1$.

Now, consider homomorphically evaluating P on a ciphertext of x. Suppose that every bit of x is encrypted by MKHE under a different public key, i.e., for $i = 1, \ldots, n$, $x_i \in \{0,1\}$ is encrypted under a public key pk_i of MKHE. To homomorphically evaluate P, we need to evaluate $P_v(x)$ backward from terminal nodes to v_0, which results in a ciphertext of $P_{v_0}(x) = P(x)$. Let $(v^{(\ell)}, v^{(\ell-1)}, \ldots, v^{(0)})$ for $v^{(\ell)} = v_0$ and $v^{(0)} \in T$ be a path to obtain $P(x)$. The evaluation algorithm of our circuit-private MKHE scheme computes a ciphertext of $P_{v^{(h)}}(x)$ (for $h = 1, \ldots, \ell$) depending on a ciphertext of x_i for $i = \phi_V(v^{(h)})$, which is encrypted under pk_i. Our simulator for the malicious circuit privacy also computes a ciphertext of $P_{v^{(h)}}(x)$, but independent of pk_i. To show indistinguishability between the real evaluation and simulation, we use private expandability of MKHE to make the ciphertext of $P_{v^{(h)}}(x)$ in the real evaluation independent of the encryption key.

Building Blocks. We will use the following building blocks.

- Let $\mathsf{MKHE}_{\mathsf{PE}} = (\mathsf{dSetup}_{\mathsf{PE}}, \mathsf{KG}_{\mathsf{PE}}, \mathsf{Enc}_{\mathsf{PE}}, \mathsf{Expand}_{\mathsf{PE}}, \mathsf{Eval}_{\mathsf{PE}}, \mathsf{Dec}_{\mathsf{PE}})$ be a privately-expandable MKFHE scheme with distributed setup, for which we also assume weak

circular security. We also require that this scheme is additive homomorphic over fresh (pre-expanded) ciphertexts such that for any $i \in [N]$, public parameters $\mathbf{pp} = (\mathsf{pp}_j)_{j\in[N]}$ generated by $\mathsf{dSetup}_{\mathsf{PE}}(1^\lambda, 1^N, i)$, an honestly generated key pair $(\mathsf{pk}_{\mathsf{PE},i}, \mathsf{sk}_{\mathsf{PE},i}) \in [\mathsf{KG}_{\mathsf{PE}}(\mathbf{pp}, i)]$, plaintexts $x_1, x_2 \in \{0,1\}$, and randomness r_1, r_2 for $\mathsf{Enc}_{\mathsf{PE}}$, we have $\mathsf{Enc}_{\mathsf{PE}}(\mathsf{pk}_{\mathsf{PE},i}, x_1; r_1) + \mathsf{Enc}_{\mathsf{PE}}(\mathsf{pk}_{\mathsf{PE},i}, x_2; r_2) = \mathsf{Enc}_{\mathsf{PE}}(\mathsf{pk}_{\mathsf{PE},i}, x_1 + x_2; r_1 + r_2)$. (In the following, we will just use "$-$" to denote the homomorphic subtraction between ciphertexts.)

- Let $\mathsf{SKHE}_{\mathsf{mCP}} = (\mathsf{KG}_{\mathsf{mCP}}, \mathsf{Enc}_{\mathsf{mCP}}, \mathsf{Eval}_{\mathsf{mCP}}, \mathsf{Dec}_{\mathsf{mCP}})$ be a maliciously circuit-private *single-key* FHE scheme. (see Sect. 2.3 for the formal definitions of single-key HE and malicious circuit privacy for single-key HE.)

Notation and convention Values with subscript PE (resp. mCP) will denote those related to $\mathsf{MKHE}_{\mathsf{PE}}$ (resp. $\mathsf{SKHE}_{\mathsf{mCP}}$). We will omit a subscript for pp.

To lighten the notation, we use the convention that $\mathsf{Enc}_{\mathsf{PE}}$ takes as plaintext input a bit-string x (rather than a single bit), and outputs the concatenation of the bit-wise encryption of x. Also, to easily grasp what is encrypted, we will use the notation $[\![x]\!]_i$ to mean bit-wise encryption of a bit-string x under the i-th public key $\mathsf{pk}_{\mathsf{PE},i}$ of $\mathsf{MKHE}_{\mathsf{PE}}$. (For example, we write $[\![x]\!]_i \xleftarrow{\$} \mathsf{Enc}_{\mathsf{PE}}(\mathsf{pk}_{\mathsf{PE},i}, x)$.) An expanded ciphertext of $[\![x]\!]_i$ will be denoted $\widetilde{[\![x]\!]}$.

We will use the same convention for $\mathsf{Enc}_{\mathsf{mCP}}$, and use the notation $[x]_i$ to denote an encryption of x under the i-th public key $\mathsf{pk}_{\mathsf{mCP},i}$ of $\mathsf{SKHE}_{\mathsf{mCP}}$.

Validation circuit To achieve malicious circuit privacy, the evaluation algorithm $\mathsf{Eval}_{\mathsf{BP}}$ of our scheme will homomorphically evaluate the following circuit Validate to check whether public/secret keys and ciphertexts input to $\mathsf{Eval}_{\mathsf{BP}}$ are well-formed, in which case (and only then) it outputs some hardwired value. Validate has the following values hardwired: $N, q \in \mathbb{N}$, public parameters $\mathbf{pp}$, an index $i \in [N]$, a public key $\mathsf{pk}_{\mathsf{PE}}$, q ciphertexts $\{\mathsf{ct}_{\mathsf{PE},k}\}_{k\in[q]}$, and some value $\mathsf{out} \in \{0,1\}^*$. As input, it takes a secret key $\mathsf{sk}_{\mathsf{PE}}$ and randomness r_{KG} and $\{r_{\mathsf{Enc},k}\}_{k\in[q]}$, and its output is defined as follows [9]:

$$\mathsf{Validate}^{N,q}_{\mathbf{pp},i,\mathsf{pk}_{\mathsf{PE}},\{\mathsf{ct}_{\mathsf{PE},k}\}_{k\in[q]},\mathsf{out}}(\mathsf{sk}_{\mathsf{PE}}, r_{\mathsf{KG}}, \{r_{\mathsf{Enc},k}\}_{k\in[q]})$$
$$:= \begin{cases} \mathsf{out} & \text{if } (\mathsf{pk}_{\mathsf{PE}}, \mathsf{sk}_{\mathsf{PE}}) = \mathsf{KG}_{\mathsf{PE}}(\mathbf{pp}, i; r_{\mathsf{KG}}), \\ & \text{and } \forall k \in [q], \exists x_k \in \{0,1\} : \mathsf{ct}_{\mathsf{PE},k} = \mathsf{Enc}_{\mathsf{PE}}(\mathsf{pk}_{\mathsf{PE}}, x_k; r_{\mathsf{Enc},k}) \\ 0^{|\mathsf{out}|} & \text{otherwise} \end{cases}$$

Construction Using the building blocks $\mathsf{MKHE}_{\mathsf{PE}}$ and $\mathsf{SKHE}_{\mathsf{mCP}}$, and the validation circuit Validate described above, we describe our maliciously circuit-private MKHE scheme with distributed setup for length-ℓ branching programs $\mathsf{MKHE}_{\mathsf{BP}} = (\mathsf{dSetup}_{\mathsf{BP}}, \mathsf{KG}_{\mathsf{BP}}, \mathsf{Enc}_{\mathsf{BP}}, \mathsf{Eval}_{\mathsf{BP}}, \mathsf{Dec}_{\mathsf{BP}})$ below. (For convenience, a one-page version of the description is given in Figure 1 in page 48.)

- $\mathsf{dSetup}_{\mathsf{BP}}(1^\lambda, 1^N, i \in [N])$: Output $\mathsf{pp}_i \xleftarrow{\$} \mathsf{dSetup}_{\mathsf{PE}}(1^\lambda, 1^N, i)$.
- $\mathsf{KG}_{\mathsf{BP}}(\mathbf{pp}, i)$: Pick randomness $r_{\mathsf{KG},i}$ for $\mathsf{KG}_{\mathsf{PE}}$, and compute

$$(\mathsf{pk}_{\mathsf{PE},i}, \mathsf{sk}_{\mathsf{PE},i}) := \mathsf{KG}_{\mathsf{PE}}(\mathbf{pp}, i; r_{\mathsf{KG},i}), (\mathsf{pk}_{\mathsf{mCP},i}, \mathsf{sk}_{\mathsf{mCP},i}) \xleftarrow{\$} \mathsf{KG}_{\mathsf{mCP}}(1^\lambda),$$
$$[\![\mathsf{sk}_{\mathsf{PE},i}]\!]_i \xleftarrow{\$} \mathsf{Enc}_{\mathsf{PE}}(\mathsf{pk}_{\mathsf{PE},i}, \mathsf{sk}_{\mathsf{PE},i}), [\mathsf{sk}_{\mathsf{PE},i} \| r_{\mathsf{KG},i}]_i \xleftarrow{\$} \mathsf{Enc}_{\mathsf{mCP}}(\mathsf{pk}_{\mathsf{mCP},i}, \mathsf{sk}_{\mathsf{PE},i} \| r_{\mathsf{KG},i}).$$

[9] We allow $\{\mathsf{ct}_{\mathsf{PE},k}\}_{k\in[q]}$ to be empty, in which case Validate is defined to not take $\{r_{\mathsf{Enc},k}\}$ as input, and the check of $\{\mathsf{ct}_{\mathsf{PE},k}\}$ is omitted.

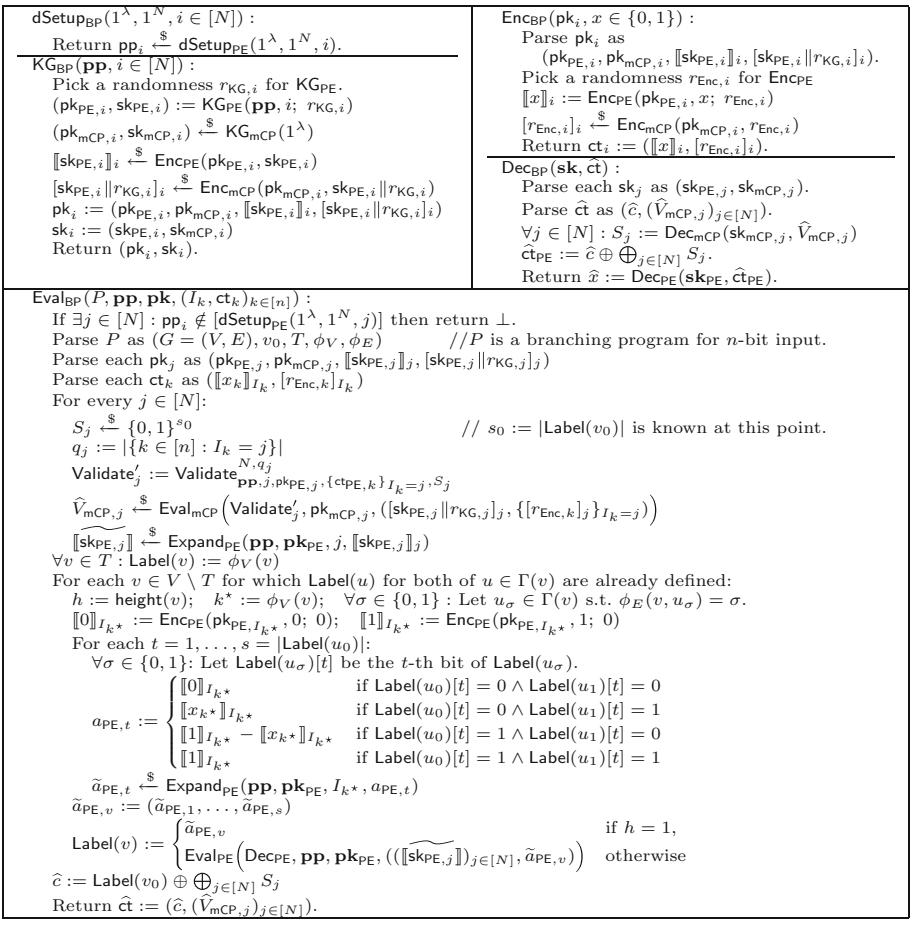

Fig. 1 Maliciously circuit private MKHE with distributed setup for branching programs $\mathsf{MKHE}_{\mathsf{BP}} = (\mathsf{dSetup}_{\mathsf{BP}}, \mathsf{KG}_{\mathsf{BP}}, \mathsf{Enc}_{\mathsf{BP}}, \mathsf{Eval}_{\mathsf{BP}}, \mathsf{Dec}_{\mathsf{BP}})$ based on privately expandable MKFHE with distributed setup $\mathsf{MKHE}_{\mathsf{PE}} = (\mathsf{dSetup}_{\mathsf{PE}}, \mathsf{KG}_{\mathsf{PE}}, \mathsf{Enc}_{\mathsf{PE}}, \mathsf{Expand}_{\mathsf{PE}}, \mathsf{Eval}_{\mathsf{PE}}, \mathsf{Dec}_{\mathsf{PE}})$ and maliciously circuit private single-key FHE $\mathsf{SKHE}_{\mathsf{mCP}} = (\mathsf{KG}_{\mathsf{mCP}}, \mathsf{Enc}_{\mathsf{mCP}}, \mathsf{Eval}_{\mathsf{mCP}}, \mathsf{Dec}_{\mathsf{mCP}})$

Output $\mathsf{pk}_i := (\mathsf{pk}_{\mathsf{PE},i}, \mathsf{pk}_{\mathsf{mCP},i}, [\![\mathsf{sk}_{\mathsf{PE},i}]\!]_i, [\mathsf{sk}_{\mathsf{PE},i} \| r_{\mathsf{KG},i}]_i)$ and $\mathsf{sk}_i := (\mathsf{sk}_{\mathsf{PE},i}, \mathsf{sk}_{\mathsf{mCP},i})$.

- $\mathsf{Enc}_{\mathsf{BP}}(\mathsf{pk}_i, x \in \{0,1\})$: Parse pk_i as $(\mathsf{pk}_{\mathsf{PE},i}, \mathsf{pk}_{\mathsf{mCP},i}, [\![\mathsf{sk}_{\mathsf{PE},i}]\!]_i, [\mathsf{sk}_{\mathsf{PE},i} \| r_{\mathsf{KG},i}])$. Pick randomness $r_{\mathsf{Enc},i}$ for $\mathsf{Enc}_{\mathsf{PE}}$, and compute

$$[\![x]\!]_i := \mathsf{Enc}_{\mathsf{PE}}(\mathsf{pk}_{\mathsf{PE},i}, x; r_{\mathsf{Enc},i}), \qquad [r_{\mathsf{Enc},i}]_i \xleftarrow{\$} \mathsf{Enc}_{\mathsf{mCP}}(\mathsf{pk}_{\mathsf{mCP},i}, r_{\mathsf{Enc},i}).$$

Output $\mathsf{ct}_i := ([\![x]\!]_i, [r_{\mathsf{Enc},i}]_i)$.

- $\mathsf{Eval}_{\mathsf{BP}}(P, \mathbf{pp}, \mathbf{pk}, (I_k, \mathsf{ct}_k)_{k\in[n]})$: If there exists $j \in [N]$ such that $\mathsf{pp}_j \notin [\mathsf{dSetup}_{\mathsf{PE}}(1^\lambda, 1^N, j)]$, then output $\perp$ and terminate. Otherwise, parse P as a length-ℓ branching program $(G = (V, E), v_0, T, \phi_V, \phi_E)$, and suppose its input length is n-bit. Parse each pk_j as $(\mathsf{pk}_{\mathsf{PE},j}, \mathsf{pk}_{\mathsf{mCP},j}, [\![\mathsf{sk}_{\mathsf{PE},j}]\!]_j, [\mathsf{sk}_{\mathsf{PE},j} \| r_{\mathsf{KG},j}]_j)$ and each ct_k as $([\![x_k]\!]_{I_k}, [r_{\mathsf{Enc},k}]_{I_k})$. Let s_0 be the length of $\mathsf{Label}(v_0)$ determined below. (s_0 itself is known a priori.) For every $j \in [N]$, choose $S_j \xleftarrow{\$} \{0,1\}^{s_0}$, set $q_j := |\{k \in [n] : I_k = j\}|$, and

compute

$$\widehat{V}_{\mathsf{mCP},j} \xleftarrow{\$} \mathsf{Eval}_{\mathsf{mCP}}\left(\begin{array}{l}\mathsf{Validate}^{N,q_j}_{\mathbf{pp},j,\mathsf{pk}_{\mathsf{PE},j},\{[\![x_k]\!]_{I_k}\}_{I_k=j},S_j},\ \mathsf{pk}_{\mathsf{mCP},j},\\ ([\mathsf{sk}_{\mathsf{PE},j}\|r_{\mathsf{KG},j}]_j,\ \{[r_{\mathsf{Enc},k}]_j\}_{I_k=j})\end{array}\right),$$

$$\widetilde{[\![\mathsf{sk}_{\mathsf{PE},j}]\!]} \xleftarrow{\$} \mathsf{Expand}_{\mathsf{PE}}(\mathbf{pp}, \mathbf{pk}_{\mathsf{PE}}, j, [\![\mathsf{sk}_{\mathsf{PE},j}]\!]_j).$$

For each $v \in T$, set $\mathsf{Label}(v) := \phi_V(v)$. For each [10] $v \in V \setminus T$ for which $\mathsf{Label}(u_0)$ and $\mathsf{Label}(u_1)$ with $\Gamma(v) = \{u_0, u_1\}$ are already defined, do the following:

- Let $h := \mathsf{height}(v)$, $k^\star := \phi_V(v)$, and $\{u_0, u_1\} = \Gamma(v)$ such that $\phi_E(v, u_0) = 0$ and $\phi_E(v, u_1) = 1$.
- Let $[\![0]\!]_{I_{k^\star}} := \mathsf{Enc}_{\mathsf{PE}}(\mathsf{pk}_{\mathsf{PE},I_{k^\star}}, 0; 0)$ and $[\![1]\!]_{I_{k^\star}} := \mathsf{Enc}_{\mathsf{PE}}(\mathsf{pk}_{\mathsf{PE},I_{k^\star}}, 1; 0)$.
- For $t = 1, \ldots, s = |\mathsf{Label}(u_0)|$, do the following:
  - For $\sigma \in \{0, 1\}$, let $\mathsf{Label}(u_\sigma)[t]$ be the t-th bit of $\mathsf{Label}(u_\sigma)$.
  - Set

$$a_{\mathsf{PE},t} := \begin{cases} [\![0]\!]_{I_{k^\star}} & \text{if } \mathsf{Label}(u_0)[t] = 0 \wedge \mathsf{Label}(u_1)[t] = 0 \\ [\![x_{k^\star}]\!]_{I_{k^\star}} & \text{if } \mathsf{Label}(u_0)[t] = 0 \wedge \mathsf{Label}(u_1)[t] = 1 \\ [\![1]\!]_{I_{k^\star}} - [\![x_{k^\star}]\!]_{I_{k^\star}} & \text{if } \mathsf{Label}(u_0)[t] = 1 \wedge \mathsf{Label}(u_1)[t] = 0 \\ [\![1]\!]_{I_{k^\star}} & \text{if } \mathsf{Label}(u_0)[t] = 1 \wedge \mathsf{Label}(u_1)[t] = 1 \end{cases},$$

  and compute $\widetilde{a}_{\mathsf{PE},t} \xleftarrow{\$} \mathsf{Expand}_{\mathsf{PE}}(\mathbf{pp}, \mathbf{pk}_{\mathsf{PE}}, I_{k^\star}, a_{\mathsf{PE},t})$. [11]
- Set $\widetilde{a}_{\mathsf{PE},v} := (\widetilde{a}_{\mathsf{PE},1}, \ldots, \widetilde{a}_{\mathsf{PE},s})$. [12]
- Compute

$$\mathsf{Label}(v) := \begin{cases} \widetilde{a}_{\mathsf{PE},v} & \text{if } h = 1, \\ \mathsf{Eval}_{\mathsf{PE}}\left(\mathsf{Dec}_{\mathsf{PE}}, \mathbf{pp}, \mathbf{pk}_{\mathsf{PE}}, ((\widetilde{[\![\mathsf{sk}_{\mathsf{PE},j}]\!]})_{j\in[N]}, \widetilde{a}_{\mathsf{PE},v})\right) & \text{otherwise} \end{cases}.$$

  where the inputs to $\mathsf{Dec}_{\mathsf{PE}}$ in $\mathsf{Eval}_{\mathsf{PE}}$ are naturally arranged: the secret keys encrypted in $(\widetilde{[\![\mathsf{sk}_{\mathsf{PE},j}]\!]})_{j\in[N]}$ are used as $\mathbf{sk}_{\mathsf{PE}}$, and what is encrypted in $\widetilde{a}_{\mathsf{PE},v}$ (which, for honestly generated ciphertexts, is $\mathsf{Label}(u_{x_{k^\star}})$) is used as an expanded ciphertext to be decrypted.

Finally, output $\widehat{\mathsf{ct}} := (\mathsf{Label}(v_0) \oplus \bigoplus_{j\in[N]} S_j, (\widehat{V}_{\mathsf{mCP},j})_{j\in[N]})$.

- $\mathsf{Dec}_{\mathsf{BP}}(\mathbf{sk}, \widehat{\mathsf{ct}})$: Parse each sk_j as $(\mathsf{sk}_{\mathsf{PE},j}, \mathsf{sk}_{\mathsf{mCP},j})$ and $\widehat{\mathsf{ct}}$ as $(\widehat{c}, (\widehat{V}_{\mathsf{mCP},j})_{j\in[N]})$. For every $j \in [N]$, compute $S_j := \mathsf{Dec}_{\mathsf{mCP}}(\mathsf{sk}_{\mathsf{mCP},j}, \widehat{V}_{\mathsf{mCP},j})$. Let $\widehat{\mathsf{ct}}_{\mathsf{PE}} := \widehat{c} \oplus \bigoplus_{j\in[N]} S_j$. Output $\widehat{x} := \mathsf{Dec}_{\mathsf{PE}}(\mathbf{sk}_{\mathsf{PE}}, \widehat{\mathsf{ct}}_{\mathsf{PE}})$.

Correctness and security We now see the correctness, security, and malicious circuit privacy of $\mathsf{MKHE}_{\mathsf{BP}}$.

Theorem 4.1 (Correctness) *$\mathsf{MKHE}_{\mathsf{BP}}$ satisfies correctness.*

[10] This step will be performed from the nodes in the last layer to the initial node v_0, so that $\mathsf{Label}(v)$ for every $v \in V \setminus T$ will be defined in the end.

[11] The way we compute $\widetilde{a}_{\mathsf{PE},t}$ is different from [17]. In their construction, $\widetilde{a}_{\mathsf{PE},t}$ in the case $\mathsf{Label}(u_0)[t] = \mathsf{Label}(u_1)[t] = 1$ is computed deterministically from the values that are generated outside the loop regarding t. This seems to make $\widetilde{a}_{\mathsf{PE},t}$ distinguishable from what the simulator for the (malicious) circuit privacy in the proof generates. Our design of $\mathsf{Eval}_{\mathsf{BP}}$ simplifies the design of the homomorphic evaluation algorithm in [17] while also removing this bug.

[12] One can check that each $\widetilde{a}_{\mathsf{PE},t}$ is an expanded ciphertext of $\mathsf{Label}(u_{x_{k^\star}})[t]$. Hence, $\widetilde{a}_{\mathsf{PE},v}$ is bit-wise expanded ciphertexts of $\mathsf{Label}(u_{x_{k^\star}})$.

Proof Let $\mathbf{pp}$, $\mathbf{pk}$, and $\mathbf{sk}$ be honestly generated public parameters, public keys, and secret keys, respectively. Let $I_1, \ldots, I_n \in [N]$ be arbitrary indices, $x = (x_1, \ldots, x_n) \in \{0,1\}^n$ be arbitrary plaintexts, and $\mathsf{ct}_k = (\llbracket x_k \rrbracket_{I_k}, [r_{\mathsf{Enc},I_k}]_{I_k}) \xleftarrow{\$} \mathsf{Enc}_{\mathsf{BP}}(\mathsf{pk}_{I_k}, x_k)$ for every $k \in [n]$. Let $P = (G = (V, E), v_0, T, \phi_V, \phi_E)$ be a length-ℓ branching program. Now, consider executing $\widehat{\mathsf{ct}} = (\widehat{c}, (\widehat{V}_{\mathsf{mCP},j})_{j\in[N]}) \xleftarrow{\$} \mathsf{Eval}_{\mathsf{BP}}(P, \mathbf{pp}, \mathbf{pk}, (I_k, \mathsf{pk}_k)_{k\in[n]})$ and $\mathsf{Dec}_{\mathsf{BP}}(\mathbf{sk}, \widehat{\mathsf{ct}})$. Since all public keys $\mathbf{pk}$ and ciphertexts $(\mathsf{ct}_k)_{k\in[n]}$ are honestly generated, the check regarding $\mathsf{pk}_{\mathsf{PE},j}$ and $\{\llbracket x_k \rrbracket_{I_k}\}_{I_k=j}$ in the homomorphic evaluation of $\mathsf{Validate}$ done by $\mathsf{Eval}_{\mathsf{mCP}}$ never fails, and every $\widehat{V}_{\mathsf{mCP},j}$ decrypts to S_j. This implies $\widehat{\mathsf{ct}}_{\mathsf{PE}} = \widehat{c} \oplus \bigoplus_{j\in[N]} S_j = \mathsf{Label}(v_0)$. Hence, to show the correctness, it is sufficient to show $\mathsf{Dec}_{\mathsf{PE}}(\mathbf{sk}_{\mathsf{PE}}, \mathsf{Label}(v_0)) = P(x)$.

For each $h \in \{0\} \cup [\ell]$, let $v^{(h)} \in V$ be a node such that $\mathsf{height}(v^{(h)}) = h$ and $\Gamma(v^{(h)}) = \{v_0^{(h-1)}, v_1^{(h-1)}\}$, and thus $v_0 = v^{(\ell)}$. Intuitively, $\mathsf{Eval}_{\mathsf{BP}}$ homomorphically computes the branching program P backwards from the terminal nodes to the initial node: Every layer of the branching program computes $P_{v^{(h)}}(x) = P_{v^{(h-1)}_{x_{k^\star(h)}}}(x)$ where $k^{\star(h)} := \phi_V(v^{(h)})$. We now inductively show that for every $h \in [\ell]$, every vertex $v^{(h)}$ at height h satisfies the property that $\mathsf{Label}(v^{(h)})$ is a bit-wise expanded ciphertexts of $P_{v^{(h)}}(x)$, and thus $\mathsf{Dec}_{\mathsf{PE}}(\mathbf{sk}_{\mathsf{PE}}, \mathsf{Label}(v^{(h)})) = P_{v^{(h)}}(x)$ holds.

Base case h = 1 : by the construction of $\mathsf{Eval}_{\mathsf{BP}}$, $\mathsf{Label}(v^{(1)}) = \widetilde{a}_{\mathsf{PE},v^{(1)}}$ is bit-wise expanded ciphertexts of $\mathsf{Label}(v^{(0)}_{x_{k^\star(1)}}) = x_{k^\star(1)} = P_{v^{(1)}}(x)$ [13].

This means that

$$\mathsf{Dec}_{\mathsf{PE}}(\mathbf{sk}_{\mathsf{PE}}, \mathsf{Label}(v^{(1)})) = P_{v^{(1)}}(x).$$

Assumption for the induction (case h − 1) : suppose as the assumption for the induction that the case $h-1$ is true, i.e. every $v^{(h-1)}$ at height $h-1$ satisfies the property that $\mathsf{Label}(v^{(h-1)})$ is bit-wise expanded ciphertexts of $P_{v^{(h-1)}}(x)$ and we have

$$\mathsf{Dec}_{\mathsf{PE}}(\mathbf{sk}_{\mathsf{PE}}, \mathsf{Label}(v^{(h-1)})) = P_{v^{(h-1)}}(x).$$

Case h : let $v^{(h)}$ be an arbitrary vertex at height h, and let $\Gamma(v^{(h)}) = \{v_0^{(h-1)}, v_1^{(h-1)}\}$ with $\phi_E(v^{(h)}, v_\sigma^{(h-1)}) = \sigma$ for $\sigma \in \{0,1\}$. Recall that by the construction of $\mathsf{Eval}_{\mathsf{BP}}$, $\widetilde{a}_{\mathsf{PE},v^{(h)}}$ is bit-wise expanded ciphertexts of $\mathsf{Label}(v^{(h-1)}_{x_{k^\star(h)}})$, and $\mathsf{Label}(v^{(h)})$ is an output of $\mathsf{Eval}_{\mathsf{PE}}(\mathsf{Dec}_{\mathsf{PE}}, \mathbf{pp}, \mathbf{pk}_{\mathsf{PE}}, ((\widetilde{\llbracket \mathsf{sk}_{\mathsf{PE},j} \rrbracket_j})_{j\in[N]}, \widetilde{a}_{\mathsf{PE},v^{(h)}}))$. Hence, $\mathsf{Label}(v^{(h)})$ is bit-wise expanded ciphertexts of

$$\mathsf{Dec}_{\mathsf{PE}}(\mathbf{sk}_{\mathsf{PE}}, \mathsf{Label}(v^{(h-1)}_{x_{k^\star(h)}})) = P_{v^{(h-1)}_{x_{k^\star(h)}}}(x) = P_{v^{(h)}}(x),$$

which means $\mathsf{Dec}_{\mathsf{PE}}(\mathbf{sk}_{\mathsf{PE}}, \mathsf{Label}(v^{(h)})) = P_{v^{(h)}}(x)$ holds.

This is true for every vertex at height h, and thus proves the inductive step. Hence, we have $\mathsf{Dec}_{\mathsf{PE}}(\mathbf{sk}_{\mathsf{PE}}, \mathsf{Label}(v^{\ell})) = \mathsf{Dec}_{\mathsf{PE}}(\mathbf{sk}_{\mathsf{PE}}, \mathsf{Label}(v_0)) = P(x)$, and thus the scheme satisfies correctness. □

For the semantic security of $\mathsf{MKHE}_{\mathsf{BP}}$, we need to assume $\mathsf{MKHE}_{\mathsf{PE}}$ is weakly circular secure. Since the proof is straightforward, we only give a proof sketch.

Theorem 4.2 (Security) *If MKHE_{PE} is weakly circular secure and SKHE_{mCP} is semantically secure, then MKHE_{BP} is semantically secure.*

[13] We here assume that for $b \in \{0,1\}$, $v_b^{(0)} \in \Gamma(v^{(1)})$ satisfies $\phi_V(v_b^{(0)}) = b$.

Proof of Sketch By the semantic security of $\mathsf{SKHE}_{\mathsf{mCP}}$, $[\mathsf{sk}_{\mathsf{PE},i} \| r_{\mathsf{KG},i}]_i$ in the challenge public key pk_i and $[r_{\mathsf{Enc},i}]$ in the challenge ciphertext ct_i can be replaced by encryptions of unrelated messages of appropriate length without being noticed by an adversary. Then, the weak circular security of $\mathsf{MKHE}_{\mathsf{PE}}$ guarantees that the information of the challenge message x does not leak from $[\![x]\!]_i$ in ct_i. □

The following theorem guarantees the malicious circuit privacy of $\mathsf{MKHE}_{\mathsf{BP}}$. The intuition and notation for the proof of this theorem are given at the beginning of this section (Sect. 4.2).

Theorem 4.3 (Malicious Circuit Privacy) *If MKHE_{PE} is privately expandable and SKHE_{mCP} is maliciously circuit-private, then MKHE_{BP} is maliciously circuit private.*

Proof Let $\mathsf{Ext}_{\mathsf{mCP}}$ and $\mathsf{Sim}_{\mathsf{mCP}}$ be respectively the extractor and simulator that are guaranteed to exist by the malicious circuit privacy of $\mathsf{SKHE}_{\mathsf{mCP}}$. We construct an extractor $\mathsf{Ext}_{\mathsf{BP}}$ and a simulator $\mathsf{Sim}_{\mathsf{BP}}$ as follows:

- $\mathsf{Ext}_{\mathsf{BP}}(\mathbf{pp}^*, i, \mathsf{pk}_i^*, \mathsf{ct}^*)$: Parse pk_i^* as $(\mathsf{pk}_{\mathsf{PE},i}^*, \mathsf{pk}_{\mathsf{mCP},i}^*, [\![\mathsf{sk}_{\mathsf{PE},i}]\!]_i^*, [\mathsf{sk}_{\mathsf{PE},i} \| r_{\mathsf{KG},i}]_i^*)$ and ct^* as $([\![x]\!]_i^*, [r_{\mathsf{Enc}}]_i^*)$.
  Extract [14]
  $$\mathsf{sk}_{\mathsf{PE},i}^* \| r_{\mathsf{KG},i}^* := \mathsf{Ext}_{\mathsf{mCP}}(\mathsf{pk}_{\mathsf{mCP},i}^*, [\mathsf{sk}_{\mathsf{PE},i} \| r_{\mathsf{KG},i}]_i^*),$$
  $$r_{\mathsf{Enc}}^* := \mathsf{Ext}_{\mathsf{mCP}}(\mathsf{pk}_{\mathsf{mCP},i}^*, [r_{\mathsf{Enc}}]_i^*).$$
  Check whether $(\mathsf{pk}_{\mathsf{PE},i}^*, \mathsf{sk}_{\mathsf{PE},i}^*) = \mathsf{KG}_{\mathsf{PE}}(\mathbf{pp}^*, i; r_{\mathsf{KG}}^*)$ and there exists $x' \in \{0,1\}$ such that $[\![x]\!]_i^* = \mathsf{Enc}_{\mathsf{PE}}(\mathbf{pp}^*, \mathsf{pk}_{\mathsf{PE},i}^*, x'; r_{\mathsf{Enc}}^*)$. If the check passes, then output x'. Otherwise, output 0.
- $\mathsf{Sim}_{\mathsf{BP}}(\mathbf{pp}^*, \mathbf{pk}^*, (I_k, \mathsf{ct}_k^*)_{k\in[n]}, b^* \in \{0,1\})$: If there exists $j \in [N]$ such that $\mathsf{pp}_j^* \notin [\mathsf{dSetup}_{\mathsf{PE}}(1^\lambda, 1^N, j)]$, then output $\perp$ and terminate. Otherwise, parse each pk_j^* as $(\mathsf{pk}_{\mathsf{PE},j}^*, \mathsf{pk}_{\mathsf{mCP},j}^*, [\![\mathsf{sk}_{\mathsf{PE},j}]\!]_j^*, [\mathsf{sk}_{\mathsf{PE},j} \| r_{\mathsf{KG},j}]_j^*)$ and each ct_k^* as $([\![x_k]\!]_{I_k}^*, [r_{\mathsf{Enc},k}]_{I_k}^*)$. For every $j \in [N]$, do the following:
  - Sample $S_j \xleftarrow{\$} \{0,1\}^{s_0}$.
  - Do the same check on $\mathsf{pk}_{\mathsf{PE},j}^*$ and $[\![x_k]\!]_j^*$ as done in $\mathsf{Ext}_{\mathsf{BP}}(\mathbf{pp}^*, j, \mathsf{pk}_{\mathsf{PE},j}^*, \mathsf{ct}_k^*)$ for all k such that $I_k = j$.
  - Compute
  $$\widehat{V}_{\mathsf{mCP},j}^* := \begin{cases} \mathsf{Sim}_{\mathsf{mCP}}\left(\mathsf{pk}_{\mathsf{mCP},j}^*, ([\mathsf{sk}_{\mathsf{mCP},j} \| r_{\mathsf{KG},j}]_j^*, \{[r_{\mathsf{Enc},k}]_j^*\}_{I_k=j}), S_j\right) \\ \qquad\qquad \text{if the above check passes} \\ \mathsf{Sim}_{\mathsf{mCP}}\left(\mathsf{pk}_{\mathsf{mCP},j}^*, ([\mathsf{sk}_{\mathsf{mCP},j} \| r_{\mathsf{KG},j}]_j^*, \{[r_{\mathsf{Enc},k}]_j^*\}_{I_k=j}), 0^{s_0}\right) \\ \qquad\qquad \text{otherwise} \end{cases}.$$
  - Compute $\widetilde{[\![\mathsf{sk}_{\mathsf{PE},j}]\!]}^* \xleftarrow{\$} \mathsf{Expand}_{\mathsf{PE}}(\mathbf{pp}^*, \mathbf{pk}^*, [\![\mathsf{sk}_{\mathsf{PE},j}]\!]_j^*)$.

  If one of the checks during the computation of $\widehat{V}_{\mathsf{mCP},j}$ was not satisfied, then pick $\widehat{c}^* \xleftarrow{\$} \{0,1\}^{s_0}$, and terminate with output $\widehat{\mathsf{ct}}^* := (\widehat{c}^*, (\widehat{V}_{\mathsf{mCP},j}^*)_{j\in[N]})$. Otherwise (i.e. the checks did not fail), set $\mathsf{out}_0 := b^*$, and for $h = 1, \ldots, \ell$, do the following:

[14] Each $[\mathsf{sk}_{\mathsf{PE},i} \| r_{\mathsf{KG},i}]_i^*$ is actually a vector of ciphertexts. Thus, $\mathsf{Ext}_{\mathsf{mCP}}(\mathsf{pk}_{\mathsf{mCP},i}^*, [\mathsf{sk}_{\mathsf{PE},i}]_i^*)$ here means applying $\mathsf{Ext}_{\mathsf{mCP}}(\mathsf{pk}_{\mathsf{mCP},i}^*, \cdot)$ to each ciphertext in $[\mathsf{sk}_{\mathsf{PE},i}]_i^*$ and output the concatenation of the results. Same for $\mathsf{Ext}_{\mathsf{mCP}}(\mathsf{pk}_{\mathsf{mCP},i}^*, [r_{\mathsf{KG},i}]_i^*)$ and $\mathsf{Ext}_{\mathsf{mCP}}(\mathsf{pk}_{\mathsf{mCP},i}^*, [r_{\mathsf{Enc}}]_i^*)$.

– Let $[\![0]\!]_1 := \mathsf{Enc}_{\mathsf{PE}}(\mathsf{pk}^*_{\mathsf{PE},1}, 0; 0)$ and $[\![1]\!]_1 := \mathsf{Enc}_{\mathsf{PE}}(\mathsf{pk}^*_{\mathsf{PE},1}, 1; 0)$.
– For $t = 1, \ldots, s := |\mathsf{out}_{h-1}|$ (where $|\mathsf{out}_0| := 1$), set

$$a^*_{\mathsf{PE},t} := \begin{cases} [\![0]\!]_1 & \text{if the } t\text{-th bit of } \mathsf{out}_{h-1} \text{ is } 0 \\ [\![1]\!]_1 & \text{if the } t\text{-th bit of } \mathsf{out}_{h-1} \text{ is } 1 \end{cases},$$

and compute $\widetilde{a}^*_{\mathsf{PE},t} \xleftarrow{\$} \mathsf{Expand}_{\mathsf{PE}}(\mathbf{pp}^*, \mathbf{pk}^*_{\mathsf{PE}}, 1, a^*_{\mathsf{PE},t})$.
– Set $\widetilde{a}^*_{\mathsf{PE},h} := (\widetilde{a}^*_{\mathsf{PE},1}, \ldots, \widetilde{a}^*_{\mathsf{PE},s})$.
– Compute

$$\mathsf{out}_h := \begin{cases} \widetilde{a}^*_{\mathsf{PE},1} & \text{if } h = 1, \\ \mathsf{Eval}_{\mathsf{PE}}\Big(\mathsf{Dec}_{\mathsf{PE}}, \mathbf{pp}^*, \mathbf{pk}^*_{\mathsf{PE}}, (\widetilde{[\![\mathsf{sk}_{\mathsf{PE},j}]\!]}^*)_{j\in[N]}, \widetilde{a}^*_{\mathsf{PE},h}\Big) & \text{otherwise} \end{cases}.$$

Set $\widehat{c}^* := \mathsf{out}_\ell \oplus \bigoplus_{j\in[N]} S_j$ and output $\widehat{\mathsf{ct}}^* := (\widehat{c}^*, (\widehat{V}^*_{\mathsf{mCP},j})_{j\in[N]})$.

Consider an arbitrary branching program $P = (G = (V, E), v_0, T, \phi_V, \phi_E)$ (with length ℓ, input-length n), and public parameters $\mathbf{pp}^*$, public keys $\mathbf{pk}^* = (\mathsf{pk}^*_{\mathsf{PE},j}, [\![\mathsf{sk}_{\mathsf{PE},j}]\!]^*_j, [\mathsf{sk}_{\mathsf{PE},j} \| r_{\mathsf{KG},j}]^*_j)_{j\in[N]}$, and index/ciphertext pairs $(I_k, \mathsf{ct}^*_k)_{k\in[n]}$ where $\mathsf{ct}^*_k = ([\![x_k]\!]^*_{I_k}, [r_{\mathsf{Enc},k}]^*_{I_k})$ for each $k \in [n]$. (Here, possibly malformed values are denoted with an asterisk.) Let

$$\widehat{\mathsf{ct}} = (\widehat{c}, (\widehat{V}_{\mathsf{mCP},j})_{j\in[N]}) \xleftarrow{\$} \mathsf{Eval}_{\mathsf{BP}}(P, \mathbf{pp}^*, \mathbf{pk}^*, (I_k, \mathsf{ct}^*_k)_{k\in[n]}),$$

$$\widehat{\mathsf{ct}}^* = (\widehat{c}^*, (\widehat{V}^*_{\mathsf{mCP},j})_{j\in[N]}) \xleftarrow{\$} \mathsf{Sim}_{\mathsf{BP}}(\mathbf{pp}^*, \mathbf{pk}^*, (I_k, \mathsf{ct}^*_k)_{k\in[n]}, b^*),$$

where $b^* := P(x^*_1, \ldots, x^*_n)$ and $x^*_k := \mathsf{Ext}_{\mathsf{BP}}(\mathbf{pp}^*, I_k, \mathsf{pk}^*_{I_k}, \mathsf{ct}^*_k)$ for all $k \in [n]$. Let $\mathsf{sk}^*_{\mathsf{PE},j} \| r^*_{\mathsf{KG},j} := \mathsf{Ext}_{\mathsf{mCP}}(\mathsf{pk}^*_{\mathsf{mCP},j}, [\mathsf{sk}_{\mathsf{PE},j} \| r_{\mathsf{KG},j}]^*_j)$ for all $j \in [N]$, and $r^*_{\mathsf{Enc},k} := \mathsf{Ext}_{\mathsf{mCP}}(\mathsf{pk}^*_{\mathsf{mCP},I_k}, [r_{\mathsf{Enc},k}]^*_{I_k})$ for all $k \in [n]$. What we need to show is $\widehat{\mathsf{ct}} \approx_s \widehat{\mathsf{ct}}^*$.

Firstly, it immediately follows that $(\widehat{V}_{\mathsf{mCP},j})_{j\in[N]} \approx_s (\widehat{V}^*_{\mathsf{mCP},j})_{j\in[N]}$ holds by the malicious circuit privacy of $\mathsf{SKHE}_{\mathsf{mCP}}$. Hence, we have $(\widehat{c}, (\widehat{V}_{\mathsf{mCP},j})_{j\in[N]}) \approx_s (\widehat{c}, (\widehat{V}^*_{\mathsf{mCP},j})_{j\in[N]})$, where the right hand side is the "hybrid" distribution in which $\widehat{c}$ is generated as in $\mathsf{Eval}_{\mathsf{BP}}$ and $(\widehat{V}^*_{\mathsf{mCP},j})_{j\in[N]}$ is generated as in $\mathsf{Sim}_{\mathsf{BP}}$. Furthermore, if there exists $j \in [N]$ under which

$$\mathsf{Validate}^{N,q_j}_{\mathbf{pp}^*, j, \mathsf{pk}^*_{\mathsf{PE},j}, \{[\![x_k]\!]^*_j\}_{I_k=j}, S_j}(\mathsf{sk}^*_{\mathsf{PE},j}, r^*_{\mathsf{KG},j}, \{r^*_{\mathsf{Enc},k}\}_{I_k=j}) = 0^{s_0}$$

holds, then $\widehat{V}^*_j$ is independent of S_j, and makes $\widehat{c}$ in the hybrid distribution uniformly random and independent of other values, which is exactly how $\widehat{c}^*$ in this case is generated. This means that $\widehat{\mathsf{ct}} \approx_s \widehat{\mathsf{ct}}^*$ holds in this case.

We now consider the distributions of $\widehat{c}$ and $\widehat{c}^*$ in the remaining case (i.e. the check in $\mathsf{Validate}$ is satisfied and it outputs S_j for all $j \in [N]$). In this case, it is guaranteed that every $\mathsf{pk}^*_{\mathsf{PE},j}$ is a possible output of $\mathsf{KG}_{\mathsf{PE}}(\mathbf{pp}^*, j)$, and every $[\![x_k]\!]^*_{I_k}$ is a possible output of $\mathsf{Enc}_{\mathsf{PE}}(\mathsf{pk}^*_{\mathsf{PE},I_k}, x^*_k)$. Hence, we write $[\![x^*_k]\!]_{I_k}$ instead of $[\![x_k]\!]^*_{I_k}$ to reflect this.

Let $v^{(h)} \in V$ be the vertex at height $h \in \{0\} \cup [\ell]$ along the path indicated by $(x^*_1, \ldots, x^*_n) \in \{0,1\}^n$. To show $\widehat{c} \approx_s \widehat{c}^*$, we inductively show that $\mathsf{Label}(v^{(h)})$ in $\mathsf{Eval}_{\mathsf{BP}}$ and out_h in $\mathsf{Sim}_{\mathsf{BP}}$ are statistically indistinguishable for every $h \in \{0\} \cup [\ell]$.

Base case $h = 0$: We have $\mathsf{out}_0 = b^* = \mathsf{Label}(v^{(0)})$ by the design of $\mathsf{Sim}_{\mathsf{BP}}$, and thus clearly $\mathsf{out}_0 \approx_s \mathsf{Label}(v^{(0)})$.

Assumption for the induction (case $h - 1$) : Now, suppose as an hypothesis for the induction that $\mathsf{out}_{h-1} \approx_s \mathsf{Label}(v^{(h-1)})$ holds.

Case h : For every $t = 1, \ldots, s := |\mathsf{Label}(v^{(h)})|$, if the t-th bit of out_{h-1} is 0, then $\mathsf{Sim}_{\mathsf{BP}}$ computes $\widetilde{a}^*_{\mathsf{PE},t} \xleftarrow{\$} \mathsf{Expand}_{\mathsf{PE}}(\mathbf{pp}^*, \mathbf{pk}^*_{\mathsf{PE}}, 1, [\![0]\!]_1)$. On the other hand, if $\mathsf{Label}(v^{(h-1)})[t] = 0$, then letting $k^\star = \phi_V(v^{(h)})$, $\mathsf{Eval}_{\mathsf{BP}}$ computes $\widetilde{a}_{\mathsf{PE},t} \xleftarrow{\$} \mathsf{Expand}_{\mathsf{PE}}(\mathbf{pp}^*, \mathbf{pk}^*_{\mathsf{PE}}, I_{k^\star}, a_{\mathsf{PE},t})$, where

$$a_{\mathsf{PE},t} = \begin{cases} [\![0]\!]_{I_{k^\star}} \text{ or } [\![x^*_{k^\star}]\!]_{I_{k^\star}} & \text{if } x^*_{k^\star} = \phi_E(v^{(h)}, v^{(h-1)}) = 0 \\ [\![1]\!]_{I_{k^\star}} - [\![x^*_{k^\star}]\!]_{I_{k^\star}} & \text{if } x^*_{k^\star} = \phi_E(v^{(h)}, v^{(h-1)}) = 1 \end{cases}.$$

Since at this point $[\![x^*_{k^\star}]\!]_{I_{k^\star}}$ is guaranteed to be a valid encryption of $x^*_{k^\star}$, $a_{\mathsf{PE},t}$ is guaranteed to be a valid encryption of 0 (under $\mathsf{pk}^*_{\mathsf{PE},I_{k^\star}}$). Hence, $\widetilde{a}_{\mathsf{PE},t}$ is an expanded ciphertext of 0. Thus, if the t-th bit of out_{h-1} is equal to $0 = \mathsf{Label}(v^{(h-1)})[t]$, then by the private expandability of $\mathsf{MKHE}_{\mathsf{PE}}$, we have $\widetilde{a}^*_{\mathsf{PE},t} \approx_s \widetilde{a}_{\mathsf{PE},t}$.

Similarly, if the t-th bit of out_{h-1} is 1, then $\mathsf{Sim}_{\mathsf{BP}}$ computes $\widetilde{a}^*_{\mathsf{PE},t} \xleftarrow{\$} \mathsf{Expand}_{\mathsf{PE}}(\mathbf{pp}^*, \mathbf{pk}^*_{\mathsf{PE}}, 1, [\![1]\!]_1)$. On the other hand, if $\mathsf{Label}(v^{(h-1)})[t] = 1$, then $\mathsf{Eval}_{\mathsf{BP}}$ computes $\widetilde{a}_{\mathsf{PE},t} \xleftarrow{\$} \mathsf{Expand}_{\mathsf{PE}}(\mathbf{pp}^*, \mathbf{pk}^*_{\mathsf{PE}}, I_{k^\star}, a_{\mathsf{PE},t})$, where

$$a_{\mathsf{PE},t} = \begin{cases} [\![1]\!]_{I_{k^\star}} - [\![x^*_{k^\star}]\!]_{I_{k^\star}} & \text{if } x^*_{k^\star} = \phi_E(v^{(h)}, v^{(h-1)}) = 0 \\ [\![x^*_{k^\star}]\!]_{I_{k^\star}} \text{ or } [\![1]\!]_{I_{k^\star}} & \text{if } x^*_{k^\star} = \phi_E(v^{(h)}, v^{(h-1)}) = 1 \end{cases}.$$

Thus, if the t-th bit of out_{h-1} is equal to $1 = \mathsf{Label}(v^{(h-1)})[t]$, then $a_{\mathsf{PE},t}$ is a valid encryption of 1 (under $\mathsf{pk}^*_{I_k}$). Again by the private expandability of $\mathsf{MKHE}_{\mathsf{PE}}$, we have $\widetilde{a}^*_{\mathsf{PE},t} \approx_s \widetilde{a}_{\mathsf{PE},t}$.

Averaging over $\mathsf{out}_{h-1} \approx_s \mathsf{Label}(v^{(h-1)})$, we have $\widetilde{a}^*_{\mathsf{PE},h} \approx_s \widetilde{a}_{\mathsf{PE},v^{(h)}}$.

By applying $\mathsf{Eval}_{\mathsf{PE}}(\mathsf{Dec}_{\mathsf{PE}}, \mathbf{pp}^*, \mathbf{pk}^*_{\mathsf{PE}}, (\widetilde{[\![\mathsf{sk}_{\mathsf{PE},j}]\!]}^*)_{j\in[N]}, \cdot)$ to both sides of $\widetilde{a}^*_{\mathsf{PE},h} \approx_s \widetilde{a}_{\mathsf{PE},v^{(h)}}$, we have $\mathsf{out}_h \approx_s \mathsf{Label}(v^{(h)})$, proving the inductive step. Hence, we have $\mathsf{out}_\ell \approx_s \mathsf{Label}(v^{(\ell)}) = \mathsf{Label}(v_0)$ and thus $\widehat{c}^* \approx_s \widehat{c}$.

We have shown that in any case, $(\widehat{c}, (\widehat{V}_{\mathsf{mCP},j})_{j\in[N]}) \approx_s (\widehat{c}^*, (\widehat{V}^*_{\mathsf{mCP},j})_{j\in[N]})$ holds. This means that $\mathsf{MKHE}_{\mathsf{BP}}$ satisfies malicious circuit privacy. □

Instantiation For the underlying privately expandable MKFHE scheme with distributed setup, we can use $\mathsf{MKHE}_{\mathsf{pBHP}}$ shown in Sect. 3.3, whose weak circular security directly follows from that of the original scheme $\mathsf{MKHE}_{\mathsf{BHP}}$.

For the underlying maliciously circuit-private single-key FHE scheme, we can rely on the existing results [9, 27, 36]. Specifically, Ostrovsky et al. [36] constructed a maliciously circuit-private single-key FHE scheme from the combination of compact single-key FHE and single-key HE satisfying a weak form of malicious circuit privacy (called input privacy), and the latter can be based on a statistically sender-private (2-message) OT protocol and an information-theoretic randomized encoding [27]. Brakerski and Döttling [9] recently constructed such an OT protocol based on LWE, and the MKFHE scheme $\mathsf{MKHE}_{\mathsf{BHP}}$ can be used as single-key FHE, we can realize a maliciously circuit-private single-key FHE scheme based on LWE and the weak circular security of $\mathsf{MKHE}_{\mathsf{BHP}}$.

Using the above two schemes in $\mathsf{MKHE}_{\mathsf{BP}}$, we obtain a maliciously circuit-private MKHE with distributed setup for branching programs based on LWE and the weak circular security of $\mathsf{MKHE}_{\mathsf{BHP}}$, yielding a proof of Theorem 1.1.

4.3 Achieving full homomorphism

We follow the framework of [17] that generically transforms an MKHE scheme for branching programs to a fully homomorphic one by using a (non-circuit-private) MKFHE scheme with distributed setup as an additional building block. As in our scheme for branching programs in Sect. 4.2, the main difference from [17] is that in the homomorphic evaluation algorithm, we have to make sure the public parameters belong to the support of the distributed setup algorithm.

Intuition of our construction To construct a fully homomorphic scheme, we combine the malicious circuit-private MKHE scheme $\mathsf{MKHE}_{\mathsf{BP}}$ for branching programs in Sect. 4.2 and a standard MKFHE scheme $\mathsf{MKHE}_{\mathsf{F}}$. Inputs to a function are encrypted by $\mathsf{MKHE}_{\mathsf{F}}$ and homomorphic evaluation is done by the evaluation algorithm of $\mathsf{MKHE}_{\mathsf{F}}$. The result of the homomorphic evaluation can be transformed into the ciphertext of $\mathsf{MKHE}_{\mathsf{BP}}$ in a bootstrapping-like manner. Our homomorphic evaluation algorithm also uses $\mathsf{MKHE}_{\mathsf{BP}}$ to check whether the inputs to the homomorphic evaluation are generated properly.

The homomorphic evaluation algorithm of our construction computes an output by homomorphically evaluating a composition of the decryption of $\mathsf{MKHE}_{\mathsf{BP}}$ and function that checks all the input validations success. The composite function is parametrized by the results of the homomorphic evaluation and input validations, which are the ciphertexts of $\mathsf{MKHE}_{\mathsf{BP}}$. Since the simulator for our construction can generate simulated results of the homomorphic computation by using the simulator for $\mathsf{MKHE}_{\mathsf{BP}}$, the composite function computed by our simulator is statistically indistinguishable from the one in the real evaluation.

Building blocks We will use the following building blocks.

- Let $\mathsf{MKHE}_{\mathsf{F}} = (\mathsf{dSetup}_{\mathsf{F}}, \mathsf{KG}_{\mathsf{F}}, \mathsf{Enc}_{\mathsf{F}}, \mathsf{Eval}_{\mathsf{F}}, \mathsf{Dec}_{\mathsf{F}})$ be a (non-circuit-private) MKFHE scheme with distributed setup whose decryption circuit can be computed by NC^1 circuits.
- Let $\mathsf{MKHE}_{\mathsf{BP}} = (\mathsf{dSetup}_{\mathsf{BP}}, \mathsf{KG}_{\mathsf{BP}}, \mathsf{Enc}_{\mathsf{BP}}, \mathsf{Eval}_{\mathsf{BP}}, \mathsf{Dec}_{\mathsf{BP}})$ be a maliciously circuit-private MKHE scheme with distributed setup for length-ℓ branching programs, where ℓ is a polynomial of λ large enough so that it can compute the validation circuits $\mathsf{KValidate}$ and $\mathsf{CValidate}$ introduced below, and the decryption circuit of $\mathsf{Dec}_{\mathsf{F}}$. For notational convenience, we treat this scheme as an MKHE scheme for circuits that can be computed by length ℓ-branching programs. (Thus, $\mathsf{Eval}_{\mathsf{BP}}$ takes a circuit as input, rather than a branching program).

Notation and convention We use similar notations and conventions as in Sect. 4.2, namely, subscripts F and BP for parameters, and $\mathsf{Enc}_{\mathsf{F}}$ and $\mathsf{Enc}_{\mathsf{BP}}$ takes bit-strings as a plaintext inputs. Similarly, we allow $\mathsf{Dec}_{\mathsf{F}}$ and $\mathsf{Dec}_{\mathsf{BP}}$ to take multiple ciphertexts as input, and let their outputs be the concatenation of the decryption results. We will also use bracket notations for ciphertexts: $[\![x]\!]_i$ for a ciphertext generated as $\mathsf{Enc}_{\mathsf{F}}(\mathsf{pk}_{\mathsf{F},i}, x)$, and $[x]_i$ for a ciphertext generated as $\mathsf{Enc}_{\mathsf{BP}}(\mathsf{pk}_{\mathsf{BP},i}, x)$.

Validation circuits We introduce three types of circuits that will be used in the construction of our fully homomorphic scheme.

- $\mathsf{KValidate}$ checks whether a hardwired public key of $\mathsf{MKHE}_{\mathsf{F}}$ is well-formed. It has the following values hardwired: $N \in \mathbb{N}$, public parameters $\mathbf{pp}_{\mathsf{F}}$, an index $i \in [N]$, a public key pk_{F}, and some value $\mathsf{out} \in \{0,1\}^*$. Then, it takes a pair of (candidate) secret key sk_{F} and randomness r_{KG} as input, and its output is defined as follows:

$$\mathsf{KValidate}^{N}_{\mathbf{pp}_{\mathsf{F}},i,\mathsf{pk}_{\mathsf{F}},\mathsf{out}}(\mathsf{sk}_{\mathsf{F}}, r_{\mathsf{KG}}) := \begin{cases} \mathsf{out} & \text{if } (\mathsf{pk}_{\mathsf{F}}, \mathsf{sk}_{\mathsf{F}}) = \mathsf{KG}_{\mathsf{F}}(\mathbf{pp}_{\mathsf{F}}, i; r_{\mathsf{KG}}) \\ 0 & \text{otherwise} \end{cases}.$$

- $\mathsf{CValidate}$ checks whether the hardwired ciphertext of $\mathsf{MKHE_F}$ is well-formed. It has the following values hardwired: a public key $\mathsf{pk_F}$, a ciphertext $\mathsf{ct_F}$, and some value $\mathsf{out} \in \{0,1\}^*$. Then, it takes randomness r_{Enc} as input, and its output is defined as follows:

$$\mathsf{CValidate}_{\mathsf{pk_F},\mathsf{ct_F},\mathsf{out}}(r_{\mathsf{Enc}}) := \begin{cases} \mathsf{out} & \text{if } \exists x \in \{0,1\} : \mathsf{ct_F} = \mathsf{Enc_F}(\mathsf{pk_F}, x; r_{\mathsf{Enc}}) \\ 0 & \text{otherwise} \end{cases}.$$

- $\mathsf{CombineDec}$ checks whether the given ciphertexts of $\mathsf{MKHE_{BP}}$ are correctly decrypted to a hardwired element. It has $N, q \in \mathbb{N}$ and some value $\mathsf{out} \in \{0,1\}^*$ hardwired, takes N secret keys $\mathbf{sk}_{\mathsf{BP}} = (\mathsf{sk}_{\mathsf{BP},j})_{j\in[N]}$ and q evaluated ciphertexts $\{\widehat{\mathsf{ct}}_{\mathsf{BP},j}\}_{j\in[q]}$ as input, and its output is defined as follows:

$$\mathsf{CombineDec}^{N,q}_{\mathsf{out}}\left(\mathbf{sk}_{\mathsf{BP}}, \{\widehat{\mathsf{ct}}_{\mathsf{BP},j}\}_{j\in[q]}\right) := \begin{cases} \mathsf{out} & \text{if } \mathsf{Dec_{BP}}(\mathbf{sk}_{\mathsf{BP}}, \widehat{\mathsf{ct}}_{\mathsf{BP},j}) = \mathsf{out} \\ & \qquad \text{for all } j \in [q] \\ 0 & \text{otherwise} \end{cases}.$$

Construction Using the building blocks $\mathsf{MKHE_F}$ and $\mathsf{MKHE_{BP}}$, and the validation circuits $\mathsf{KValidate}$, $\mathsf{CValidate}$, and $\mathsf{CombineDec}$, we construct our maliciously circuit private MKFHE scheme with distributed setup $\mathsf{MKHE_{mCP}} = (\mathsf{dSetup_{mCP}}, \mathsf{KG_{mCP}}, \mathsf{Enc_{mCP}}, \mathsf{Eval_{mCP}}, \mathsf{Dec_{mCP}})$ as follows. (For convenience, a one-page version of the description is given in Figure 2in page 49.)

- $\mathsf{dSetup_{mCP}}(1^\lambda, 1^N, i \in [N])$: Compute $\mathsf{pp}_{\mathsf{BP},i} \xleftarrow{\$} \mathsf{dSetup_{BP}}(1^\lambda, 1^N, i)$ and $\mathsf{pp}_{\mathsf{F},i} \xleftarrow{\$} \mathsf{dSetup_F}(1^\lambda, 1^N, i)$. Output $\mathsf{pp}_i := (\mathsf{pp}_{\mathsf{BP},i}, \mathsf{pp}_{\mathsf{F},i})$.
- $\mathsf{KG_{mCP}}(\mathbf{pp}, i \in [N])$: Parse each pp_j as $(\mathsf{pp}_{\mathsf{BP},j}, \mathsf{pp}_{\mathsf{F},j})$. Pick randomness $r_{\mathsf{KG},i}$ for $\mathsf{KG_F}$, and compute

$$(\mathsf{pk}_{\mathsf{F},i}, \mathsf{sk}_{\mathsf{F},i}) := \mathsf{KG_F}(\mathbf{pp}_{\mathsf{F}}, i; r_{\mathsf{KG},i}), \qquad (\mathsf{pk}_{\mathsf{BP},i}, \mathsf{sk}_{\mathsf{BP},i}) \xleftarrow{\$} \mathsf{KG_{BP}}(\mathbf{pp}_{\mathsf{BP}}, i),$$
$$[\![\mathsf{sk}_{\mathsf{BP},i}]\!]_i \xleftarrow{\$} \mathsf{Enc_F}(\mathsf{pk}_{\mathsf{F},i}, \mathsf{sk}_{\mathsf{BP},i}), \qquad [\mathsf{sk}_{\mathsf{F},i}]_i \xleftarrow{\$} \mathsf{Enc_{BP}}(\mathsf{pk}_{\mathsf{BP},i}, \mathsf{sk}_{\mathsf{F},i}),$$
$$[r_{\mathsf{KG},i}]_i \xleftarrow{\$} \mathsf{Enc_{BP}}(\mathsf{pk}_{\mathsf{BP},i}, r_{\mathsf{KG},i}).$$

  Output $\mathsf{pk}_i := (\mathsf{pk}_{\mathsf{F},i}, \mathsf{pk}_{\mathsf{BP},i}, [\![\mathsf{sk}_{\mathsf{BP},i}]\!]_i, [\mathsf{sk}_{\mathsf{F},i}], [r_{\mathsf{KG},i}]_i)$ and $\mathsf{sk}_i := \mathsf{sk}_{\mathsf{F},i}$.
- $\mathsf{Enc_{mCP}}(\mathsf{pk}_i, x \in \{0,1\})$: Parse pk_i as $(\mathsf{pk}_{\mathsf{F},i}, \mathsf{pk}_{\mathsf{BP},i}, [\![\mathsf{sk}_{\mathsf{BP},i}]\!]_i, [\mathsf{sk}_{\mathsf{F},i}]_i, [r_{\mathsf{KG},i}]_i)$. Pick randomness $r_{\mathsf{Enc},i}$ for $\mathsf{Enc_F}$, and compute

$$[\![x]\!]_i := \mathsf{Enc_F}(\mathsf{pk}_{\mathsf{F},i}, x; r_{\mathsf{Enc},i}), \qquad [r_{\mathsf{Enc},i}]_i \xleftarrow{\$} \mathsf{Enc_{BP}}(\mathsf{pk}_{\mathsf{BP},i}, r_{\mathsf{Enc},i}).$$

  Output $\mathsf{ct}_i := ([\![x]\!]_i, [r_{\mathsf{Enc},i}]_i)$.
- $\mathsf{Eval_{mCP}}(C, \mathbf{pp}, \mathbf{pk}, (I_k, \mathsf{ct}_k)_{k\in[n]})$: If there exists $j \in [N]$ such that $\mathsf{pp}_j \notin [\mathsf{dSetup_{mCP}}(1^\lambda, 1^N, j)]$, then output $\perp$ and terminate. Otherwise, parse each pp_j as $(\mathsf{pp}_{\mathsf{BP},j}, \mathsf{pp}_{\mathsf{F},j})$, each pk_j as $(\mathsf{pk}_{\mathsf{F},j}, \mathsf{pk}_{\mathsf{BP},j}, [\![\mathsf{sk}_{\mathsf{BP},j}]\!]_j, [\mathsf{sk}_{\mathsf{F},j}]_j, [r_{\mathsf{KG},j}]_j)$, and each ct_k as $([\![x_k]\!]_{I_k}, [r_{\mathsf{Enc},k}]_{I_k})$. Compute

$$\widehat{\mathsf{ct}}_{\mathsf{F}} \xleftarrow{\$} \mathsf{Eval_F}\Big(C, \mathbf{pp}_{\mathsf{F}}, \mathbf{pk}_{\mathsf{F}}, (I_k, [\![x_k]\!]_{I_k})_{k\in[n]}\Big),$$
$$\widehat{\mathsf{ct}}_{\mathsf{BP}} \xleftarrow{\$} \mathsf{Eval_{BP}}\Big(\mathsf{Dec_F}(\cdot, \widehat{\mathsf{ct}}_{\mathsf{F}}), \mathbf{pp}_{\mathsf{BP}}, \mathbf{pk}_{\mathsf{BP}}, (j, [\mathsf{sk}_{\mathsf{F},j}]_j)_{j\in[N]}\Big).$$

  For every $j \in [N]$, compute

$$\widehat{\mathsf{ct}}^K_{\mathsf{BP},j} \xleftarrow{\$} \mathsf{Eval_{BP}}\left(\begin{matrix}\mathsf{KValidate}^N_{\mathbf{pp}_{\mathsf{F}},j,\mathsf{pk}_{\mathsf{F},j},\widehat{\mathsf{ct}}_{\mathsf{BP}}}, \\ \mathbf{pp}_{\mathsf{BP}}, \mathbf{pk}_{\mathsf{BP}}, (j, ([\mathsf{sk}_{\mathsf{F},j}]_j, [r_{\mathsf{KG},j}]_j))\end{matrix}\right).$$

$\mathsf{dSetup}_{\mathsf{mCP}}(1^\lambda, 1^N, i \in [N])$:
Parse each pp_j as $(\mathsf{pp}_{\mathsf{F},j}, \mathsf{pp}_{\mathsf{BP},j})$.
Pick a randomness $r_{\mathsf{KG},i}$ for KG_F.
$(\mathsf{pk}_{\mathsf{F},i}, \mathsf{sk}_{\mathsf{F},i}) := \mathsf{KG}_\mathsf{F}(\mathbf{pp}_\mathsf{F}, i;\ r_{\mathsf{KG},i})$
$(\mathsf{pk}_{\mathsf{BP},i}, \mathsf{sk}_{\mathsf{BP},i}) \xleftarrow{\$} \mathsf{KG}_{\mathsf{BP}}(\mathbf{pp}_{\mathsf{BP}}, i)$
$[\mathsf{sk}_{\mathsf{F},i}]_i \xleftarrow{\$} \mathsf{Enc}_{\mathsf{BP}}(\mathsf{pk}_{\mathsf{BP},i}, \mathsf{sk}_{\mathsf{F},i})$
$[r_{\mathsf{KG},i}]_i \xleftarrow{\$} \mathsf{Enc}_{\mathsf{BP}}(\mathsf{pk}_{\mathsf{BP},i}, r_{\mathsf{KG},i})$
$[\![\mathsf{sk}_{\mathsf{BP},i}]\!]_i \xleftarrow{\$} \mathsf{Enc}_\mathsf{F}(\mathsf{pk}_{\mathsf{F},i}, \mathsf{sk}_{\mathsf{BP},i})$
$\mathsf{pk}_i := (\mathsf{pk}_{\mathsf{F},i}, \mathsf{pk}_{\mathsf{BP},i}, [\![\mathsf{sk}_{\mathsf{BP},i}]\!]_i, [\mathsf{sk}_{\mathsf{F},i}]_i, [r_{\mathsf{KG},i}]_i)$
$\mathsf{sk}_i := \mathsf{sk}_{\mathsf{F},i}$
Return $(\mathsf{pk}_i, \mathsf{sk}_i)$.

$\mathsf{KG}_{\mathsf{mCP}}(\mathbf{pp}, i \in [N])$:
Parse each pp_j as $(\mathsf{pp}_{\mathsf{F},j}, \mathsf{pp}_{\mathsf{BP},j})$.
Pick a randomness $r_{\mathsf{KG},i}$ for KG_F.
$(\mathsf{pk}_{\mathsf{F},i}, \mathsf{sk}_{\mathsf{F},i}) := \mathsf{KG}_\mathsf{F}(\mathbf{pp}_\mathsf{F}, i;\ r_{\mathsf{KG},i})$
$(\mathsf{pk}_{\mathsf{BP},i}, \mathsf{sk}_{\mathsf{BP},i}) \xleftarrow{\$} \mathsf{KG}_{\mathsf{BP}}(\mathbf{pp}_{\mathsf{BP}}, i)$
$[\mathsf{sk}_{\mathsf{F},i}]_i \xleftarrow{\$} \mathsf{Enc}_{\mathsf{BP}}(\mathsf{pk}_{\mathsf{BP},i}, \mathsf{sk}_{\mathsf{F},i})$
$[r_{\mathsf{KG},i}]_i \xleftarrow{\$} \mathsf{Enc}_{\mathsf{BP}}(\mathsf{pk}_{\mathsf{BP},i}, r_{\mathsf{KG},i})$
$[\![\mathsf{sk}_{\mathsf{BP},i}]\!]_i \xleftarrow{\$} \mathsf{Enc}_\mathsf{F}(\mathsf{pk}_{\mathsf{F},i}, \mathsf{sk}_{\mathsf{BP},i})$
$\mathsf{pk}_i := (\mathsf{pk}_{\mathsf{F},i}, \mathsf{pk}_{\mathsf{BP},i}, [\![\mathsf{sk}_{\mathsf{BP},i}]\!]_i, [\mathsf{sk}_{\mathsf{F},i}]_i, [r_{\mathsf{KG},i}]_i)$
$\mathsf{sk}_i := \mathsf{sk}_{\mathsf{F},i}$
Return $(\mathsf{pk}_i, \mathsf{sk}_i)$.

$\mathsf{Enc}_{\mathsf{mCP}}(\mathsf{pk}_i, x \in \{0,1\})$:
Parse pk_i as
$(\mathsf{pk}_{\mathsf{F},i}, \mathsf{pk}_{\mathsf{BP},i}, [\![\mathsf{sk}_{\mathsf{BP},i}]\!]_i, [\mathsf{sk}_{\mathsf{F},i}]_i, [r_{\mathsf{KG},i}]_i)$.
Pick a randomness $r_{\mathsf{Enc},i}$ for Enc_F.
$[\![x]\!]_i := \mathsf{Enc}_\mathsf{F}(\mathsf{pk}_{\mathsf{F},i}, x;\ r_{\mathsf{Enc},i})$
$[r_{\mathsf{Enc},i}]_i \xleftarrow{\$} \mathsf{Enc}_{\mathsf{BP}}(\mathsf{pk}_{\mathsf{BP},i}, r_{\mathsf{Enc},i})$
Return $\mathsf{ct}_i := ([\![x]\!]_i, [r_{\mathsf{Enc},i}]_i)$.

$\mathsf{Dec}(\mathbf{sk}, \widehat{\mathsf{ct}})$:
Parse each sk_j as $\mathsf{sk}_{\mathsf{F},j}$.
Return $\widehat{x} := \mathsf{Dec}_\mathsf{F}(\mathbf{sk}_\mathsf{F}, \widehat{\mathsf{ct}})$.

$\mathsf{Eval}_{\mathsf{mCP}}(C, \mathbf{pp}, \mathbf{pk}, (I_k, \mathsf{ct}_k)_{k\in[n]})$:
If $\exists j \in [N] : \mathsf{pp}_i \notin [\mathsf{dSetup}_{\mathsf{mCP}}(1^\lambda, 1^N, j)]$ then return $\perp$.
Parse each pp_j as $(\mathsf{pp}_{\mathsf{F},j}, \mathsf{pp}_{\mathsf{BP},j})$.
Parse each pk_j as $(\mathsf{pk}_{\mathsf{F},j}, \mathsf{pk}_{\mathsf{BP},j}, [\![\mathsf{sk}_{\mathsf{BP},j}]\!]_j, [\mathsf{sk}_{\mathsf{F},j}]_j, [r_{\mathsf{KG},j}]_j)$.
Parse each ct_k as $([\![x_k]\!]_{I_k}, [r_{\mathsf{Enc},k}]_{I_k})$.
$\widehat{\mathsf{ct}}_\mathsf{F} \xleftarrow{\$} \mathsf{Eval}_\mathsf{F}\left(C, \mathbf{pp}_\mathsf{F}, \mathbf{pk}_\mathsf{F}, (I_k, [\![x_k]\!]_{I_k})_{k\in[n]}\right)$
$\widehat{\mathsf{ct}}_{\mathsf{BP}} \xleftarrow{\$} \mathsf{Eval}_{\mathsf{BP}}\left(\mathsf{Dec}_\mathsf{F}(\cdot, \widehat{\mathsf{ct}}_\mathsf{F}), \mathbf{pp}_{\mathsf{BP}}, \mathbf{pk}_{\mathsf{BP}}, (j, [\mathsf{sk}_{\mathsf{F},j}]_j)_{j\in[N]}\right)$
For every $j \in [N]$:
$\mathsf{KValidate}'_j := \mathsf{KValidate}^N_{\mathbf{pp}_\mathsf{F}, j, \mathsf{pk}_{\mathsf{F},j}, \widehat{\mathsf{ct}}_{\mathsf{BP}}}$
$\widehat{\mathsf{ct}}^K_{\mathsf{BP},j} \xleftarrow{\$} \mathsf{Eval}_{\mathsf{BP}}\left(\mathsf{KValidate}'_j, \mathbf{pp}_{\mathsf{BP}}, \mathbf{pk}_{\mathsf{BP}}, (j, ([\mathsf{sk}_{\mathsf{F},j}]_j, [r_{\mathsf{KG},j}]_j))\right)$
For every $k \in [n]$
$\mathsf{CValidate}'_k := \mathsf{CValidate}_{\mathsf{pk}_{\mathsf{F},I_k}, [\![x]\!]_{I_k}, \widehat{\mathsf{ct}}_{\mathsf{BP}}}$
$\widehat{\mathsf{ct}}^C_{\mathsf{BP},k} \xleftarrow{\$} \mathsf{Eval}_{\mathsf{BP}}\left(\mathsf{CValidate}'_k, \mathbf{pp}_{\mathsf{BP}}, \mathbf{pk}_{\mathsf{BP}}, [r_{\mathsf{Enc},k}]_{I_k}\right)$
$\mathsf{Dec}'(\cdot) := \mathsf{Dec}_{\mathsf{BP}}\left(\cdot, \mathsf{CombineDec}^{N,N+n}_{\widehat{\mathsf{ct}}_{\mathsf{BP}}}(\cdot, \{\widehat{\mathsf{ct}}^K_{\mathsf{BP},j}\}_{j\in[N]} \cup \{\widehat{\mathsf{ct}}^C_{\mathsf{BP},k}\}_{k\in[n]})\right)$
$\widehat{\mathsf{ct}} \xleftarrow{\$} \mathsf{Eval}_\mathsf{F}\left(\mathsf{Dec}', \mathbf{pp}_\mathsf{F}, \mathbf{pk}_\mathsf{F}, (j, [\![\mathsf{sk}_{\mathsf{BP},j}]\!]_j)_{j\in[N]}\right)$
Return $\widehat{\mathsf{ct}}$.

Fig. 2 Maliciously Circuit Private MKFHE $\mathsf{MKHE}_{\mathsf{mCP}} = (\mathsf{dSetup}_{\mathsf{mCP}}, \mathsf{KG}_{\mathsf{mCP}}, \mathsf{Enc}_{\mathsf{mCP}}, \mathsf{Eval}_{\mathsf{mCP}}, \mathsf{Dec}_{\mathsf{mCP}})$ based on maliciously circuit private MKHE for branching programs $\mathsf{MKHE}_{\mathsf{BP}} = (\mathsf{dSetup}_{\mathsf{BP}}, \mathsf{KG}_{\mathsf{BP}}, \mathsf{Enc}_{\mathsf{BP}}, \mathsf{Eval}_{\mathsf{BP}}, \mathsf{Dec}_{\mathsf{BP}})$ and (non-circuit-private) MKFHE $\mathsf{MKHE}_\mathsf{F} = (\mathsf{dSetup}_\mathsf{F}, \mathsf{KG}_\mathsf{F}, \mathsf{Enc}_\mathsf{F}.\mathsf{Eval}_\mathsf{F}, \mathsf{Dec}_\mathsf{F})$. The constructed scheme as well as the building blocks are MKHE with distributed setup

For every $k \in [n]$, compute

$$\widehat{\mathsf{ct}}^C_{\mathsf{BP},k} \xleftarrow{\$} \mathsf{Eval}_{\mathsf{BP}}\left(\begin{matrix}\mathsf{CValidate}_{\mathsf{pk}_{\mathsf{F},I_k}, [\![x_k]\!]_{I_k}, \widehat{\mathsf{ct}}_{\mathsf{BP}}},\\ \mathbf{pp}_{\mathsf{BP}}, \mathbf{pk}_{\mathsf{BP}}, (I_k, [r_{\mathsf{Enc},k}]_{I_k})\end{matrix}\right).$$

Output

$$\widehat{\mathsf{ct}} \xleftarrow{\$} \mathsf{Eval}_\mathsf{F}\left(\begin{matrix}\mathsf{Dec}_{\mathsf{BP}}(\cdot, \mathsf{CombineDec}^{N,N+n}_{\widehat{\mathsf{ct}}_{\mathsf{BP}}}(\cdot, \{\widehat{\mathsf{ct}}^K_{\mathsf{BP},j}\}_{j\in[N]} \cup \{\widehat{\mathsf{ct}}^C_{\mathsf{BP},k}\}_{k\in[n]})),\\ \mathbf{pp}_\mathsf{F}, \mathbf{pk}_\mathsf{F}, (j, [\![\mathsf{sk}_{\mathsf{BP},j}]\!]_j)_{j\in[N]}\end{matrix}\right).$$

- $\mathsf{Dec}_{\mathsf{mCP}}(\mathbf{sk}, \widehat{\mathsf{ct}})$: Parse each sk_j as $\mathsf{sk}_{\mathsf{F},j}$. Output $\widehat{x} := \mathsf{Dec}_\mathsf{F}(\mathbf{sk}_\mathsf{F}, \widehat{\mathsf{ct}})$.

Correctness and security The correctness of $\mathsf{MKHE}_{\mathsf{mCP}}$ is immediate from the correctness of MKHE_F and $\mathsf{MKHE}_{\mathsf{BP}}$.

The semantic security of $\mathsf{MKHE}_{\mathsf{mCP}}$ is not straightforward. Like the construction in [17], a public key pk_i of $\mathsf{MKHE}_{\mathsf{mCP}}$ contains an encryption of a secret key $\mathsf{sk}_{\mathsf{BP},i}$ under $\mathsf{pk}_{\mathsf{F},i}$, and encryptions of a secret key $\mathsf{sk}_{\mathsf{F},i}$ and the randomness $r_{\mathsf{KG},i}$ used to generate $\mathsf{sk}_{\mathsf{F},i}$ under $\mathsf{pk}_{\mathsf{F},i}$. Hence, semantic security does not simply follow from a straightforward combination of the semantic security and/or (weak) circular security of the individual building

blocks $\mathsf{MKHE}_{\mathsf{F}}$ and $\mathsf{MKHE}_{\mathsf{BP}}$. In order to prove the semantic security of $\mathsf{MKHE}_{\mathsf{mCP}}$, it seems unavoidable to assume that the semantic security of $\mathsf{MKHE}_{\mathsf{F}}$ and $\mathsf{MKHE}_{\mathsf{BP}}$ continue to hold even if in the semantic security games of each scheme, an adversary is additionally given the values that constitute the public parameters $\mathsf{pp}_i = (\mathsf{pp}_{\mathsf{F},i}, \mathsf{pp}_{\mathsf{BP},i})$ and a public key $\mathsf{pk}_i = (\mathsf{pk}_{\mathsf{F},i}, \mathsf{pk}_{\mathsf{BP},i}, [\![\mathsf{sk}_{\mathsf{BP},i}]\!]_i, [\mathsf{sk}_{\mathsf{F},i}]_i, [r_{\mathsf{KG},i}]_i)$ of $\mathsf{MKHE}_{\mathsf{mCP}}$. [15]

Once we admit this type of somewhat non-standard circular security assumptions on $\mathsf{MKHE}_{\mathsf{F}}$ and $\mathsf{MKHE}_{\mathsf{BP}}$, the semantic security of $\mathsf{MKHE}_{\mathsf{mCP}}$ follows straightforwardly, and thus we omit the proof.

Theorem 4.4 (Security) *If MKHE_F and MKHE_{BP} satisfy the above "joint" circular security, then MKHE_{mCP} is semantically secure.*

The following theorem guarantees the malicious circuit privacy of $\mathsf{MKHE}_{\mathsf{mCP}}$. The intuition and notation for the proof of this theorem are given at the beginning of this section (Sect. 4.3).

Theorem 4.5 *(Malicious Circuit Privacy) If MKHE_{BP} is maliciously circuit-private, then so is MKHE_{mCP}.*

The proof goes similarly to that of [16, Theorem 5.1].

Proof Let $\mathsf{Ext}_{\mathsf{BP}}$ and $\mathsf{Sim}_{\mathsf{BP}}$ be the extractor and simulator, respectively, that are guaranteed to exist by the malicious circuit privacy of $\mathsf{MKHE}_{\mathsf{BP}}$. We construct an extractor $\mathsf{Ext}_{\mathsf{mCP}}$ and a simulator $\mathsf{Sim}_{\mathsf{mCP}}$ as follows.

- $\mathsf{Ext}_{\mathsf{mCP}}(\mathbf{pp}^*, i, \mathsf{pk}_i^*, \mathsf{ct}_i^*)$: Parse each pp_j^* as $(\mathsf{pp}_{\mathsf{F},j}^*, \mathsf{pp}_{\mathsf{BP},j}^*)$, pk_i^* as $(\mathsf{pk}_{\mathsf{F},i}^*, \mathsf{pk}_{\mathsf{BP},i}^*, [\mathsf{sk}_{\mathsf{F},i}]_i^*, [r_{\mathsf{KG},i}]_i^*, [\![\mathsf{sk}_{\mathsf{BP},i}]\!]_i^*)$, and ct_i^* as $([\![x]\!]_i^*, [r_{\mathsf{Enc},i}]_i^*)$. Extract
$$\begin{aligned} \mathsf{sk}_{\mathsf{F},i}^* &:= \mathsf{Ext}_{\mathsf{BP}}(\mathbf{pp}_{\mathsf{BP}}^*, i, \mathsf{pk}_{\mathsf{BP},i}^*, [\mathsf{sk}_{\mathsf{F},i}]_i^*), \\ r_{\mathsf{KG},i}^* &:= \mathsf{Ext}_{\mathsf{BP}}(\mathbf{pp}_{\mathsf{BP}}^*, i, \mathsf{pk}_{\mathsf{BP},i}^*, [r_{\mathsf{KG},i}]_i^*), \\ r_{\mathsf{Enc},i}^* &:= \mathsf{Ext}_{\mathsf{BP}}(\mathbf{pp}_{\mathsf{BP}}^*, i, \mathsf{pk}_{\mathsf{BP},i}^*, [r_{\mathsf{Enc},i}]_i^*). \end{aligned}$$
Check whether $(\mathsf{pk}_{\mathsf{F},i}^*, \mathsf{sk}_{\mathsf{F},i}^*) = \mathsf{KG}_{\mathsf{F}}(\mathbf{pp}_{\mathsf{F}}^*, i; r_{\mathsf{KG},i}^*)$ and there exists $x' \in \{0,1\}$ such that $[\![x]\!]_i^* = \mathsf{Enc}_{\mathsf{F}}(\mathsf{pk}_{\mathsf{F},i}^*, x'; r_{\mathsf{Enc},i}^*)$. If the check passes, then output x'. Otherwise, output 0.
- $\mathsf{Sim}_{\mathsf{mCP}}(\mathbf{pp}^*, \mathbf{pk}^*, (I_k, \mathsf{ct}_k^*)_{k\in[n]}, b^*)$: If there exists $j \in [N]$ such that $\mathsf{pp}_j^* \notin [\mathsf{dSetup}_{\mathsf{mCP}}(1^\lambda, 1^N, j)]$ holds, then output $\perp$ and terminate. Otherwise, parse each pp_j^* as $(\mathsf{pp}_{\mathsf{F},j}^*, \mathsf{pp}_{\mathsf{BP},j}^*)$, each pk_j^* as $(\mathsf{pk}_{\mathsf{F},j}^*, \mathsf{pk}_{\mathsf{BP},j}^*, [\![\mathsf{sk}_{\mathsf{BP},j}^*]\!]_j, [\mathsf{sk}_{\mathsf{F},j}]_j^*, [r_{\mathsf{KG},j}]_j^*)$, and each ct_k^* as $([\![x_k]\!]_{I_k}^*, [r_{\mathsf{Enc},k}]_{I_k}^*)$. Compute $\widehat{\mathsf{ct}}_{\mathsf{BP}}^* \xleftarrow{\$} \mathsf{Sim}_{\mathsf{BP}}(\mathbf{pp}_{\mathsf{BP}}^*, \mathbf{pk}_{\mathsf{BP}}^*, (j, [\mathsf{sk}_{\mathsf{F},j}]_j^*))_{j\in[N]}, b^*)$. For every $j \in [N]$, compute
$$\begin{aligned} \mathsf{sk}_{\mathsf{F},j}^* &:= \mathsf{Ext}_{\mathsf{BP}}(\mathbf{pp}_{\mathsf{BP}}^*, j, \mathsf{pk}_{\mathsf{BP},j}^*, [\mathsf{sk}_{\mathsf{F},j}]_j^*), \\ r_{\mathsf{KG},j}^* &:= \mathsf{Ext}_{\mathsf{BP}}(\mathbf{pp}_{\mathsf{BP}}^*, j, \mathsf{pk}_{\mathsf{BP},j}^*, [r_{\mathsf{KG},j}]_j^*), \\ \mathsf{out}_j^{*K} &:= \begin{cases} \widehat{\mathsf{ct}}_{\mathsf{BP}}^* & \text{if } (\mathsf{pk}_{\mathsf{F},j}^*, \mathsf{sk}_{\mathsf{F},j}^*) = \mathsf{KG}_{\mathsf{F}}(\mathbf{pp}_{\mathsf{F}}, j; r_{\mathsf{KG},j}^*) \\ 0 & \text{otherwise} \end{cases}, \end{aligned}$$

[15] For proving the semantic security of the maliciously circuit-private MKFHE scheme by Chongchitmate and Ostrovsky [17], a similar "joint" circular security assumption on the building blocks seems necessary. In the corresponding proof of [16, Theorem 5.1] (the full version of [17]), it is written that the semantic security of their maliciously circuit-private scheme follows from the building blocks, but no explanation on how the proof is obtained is given.

$$\widehat{\mathsf{ct}}^{*K}_{\mathsf{BP},j} \xleftarrow{\$} \mathsf{Sim}_{\mathsf{BP}}\left(\mathbf{pp}^*_{\mathsf{BP}}, \mathbf{pk}^*_{\mathsf{BP}}, (j, ([\mathsf{sk}_{\mathsf{BHP},j}]^*_j, [r_{\mathsf{KG},j}]^*_j)), \mathsf{out}^{*K}_j\right).$$

For every $k \in [n]$, compute

$$r^*_{\mathsf{Enc},k} := \mathsf{Ext}_{\mathsf{BP}}(\mathbf{pp}^*_{\mathsf{BP}}, I_k, \mathsf{pk}^*_{\mathsf{BP},I_k}, [r_{\mathsf{Enc},k}]^*_{I_k}),$$

$$\mathsf{out}^{*C}_k := \begin{cases} \widehat{\mathsf{ct}}^*_{\mathsf{BP}} & \text{if } \exists x' \in \{0,1\} : [\![x_k]\!]^*_{I_k} = \mathsf{Enc}_{\mathsf{F}}(\mathsf{pk}^*_{\mathsf{F},I_k}, x'; r^*_{\mathsf{Enc},k}) \\ 0 & \text{otherwise} \end{cases},$$

$$\widehat{\mathsf{ct}}^{*C}_{\mathsf{BP},k} \xleftarrow{\$} \mathsf{Sim}_{\mathsf{BP}}\left(\mathbf{pp}^*_{\mathsf{BP}}, \mathbf{pk}^*_{\mathsf{BP}}, (I_k, [r_{\mathsf{Enc},k}]^*_k), \mathsf{out}^{*C}_k\right).$$

Output

$$\widehat{\mathsf{ct}}^* \xleftarrow{\$} \mathsf{Eval}_{\mathsf{F}}\left(\begin{array}{l} \mathsf{Dec}_{\mathsf{BP}}(\cdot, \mathsf{CombineDec}^{N,N+n}_{\widehat{\mathsf{ct}}^*_{\mathsf{BP}}}(\cdot, \{\widehat{\mathsf{ct}}^{*K}_{\mathsf{BP},j}\}_{j\in[N]} \cup \{\widehat{\mathsf{ct}}^{*C}_{\mathsf{BP},k}\}_{k\in[n]})), \\ \mathbf{pp}^*_{\mathsf{F}}, \mathbf{pk}^*_{\mathsf{F}}, (j, [\![\mathsf{sk}_{\mathsf{BP},j}]\!]^*_j)_{j\in[N]} \end{array}\right).$$

Since $\widehat{\mathsf{ct}}^*_{\mathsf{BP}}$, $(\widehat{\mathsf{ct}}^{*K}_{\mathsf{BP},j})_{j\in[N]}$, and $(\widehat{\mathsf{ct}}^{*C}_{\mathsf{BP},k})_{k\in[n]}$ computed in $\mathsf{Sim}_{\mathsf{mCP}}$ are respectively statistically indistinguishable from $\widehat{\mathsf{ct}}_{\mathsf{BP}}$, $(\widehat{\mathsf{ct}}^K_{\mathsf{BP},j})_{j\in[N]}$, and $(\widehat{\mathsf{ct}}^C_{\mathsf{BP},k})_{k\in[n]}$ computed in $\mathsf{Eval}_{\mathsf{mCP}}$ by the malicious circuit privacy of $\mathsf{MKHE}_{\mathsf{BP}}$, the distribution of $\widehat{\mathsf{ct}}^*$ output from $\mathsf{Sim}_{\mathsf{mCP}}$ is also statistically indistinguishable from the evaluated ciphertext $\widehat{\mathsf{ct}}$ output from $\mathsf{Eval}_{\mathsf{mCP}}$. □

Theorems 4.4 and 4.5 yield Theorem 1.2.

Instantiation For the underlying maliciously circuit-private MKHE for branching programs, we can just use the scheme explained in Sect. 4.2. For the underlying (non-circuit-private) MKFHE, we can use the MKFHE scheme $\mathsf{MKHE}_{\mathsf{BHP}}$.

As explained above, for the resulting scheme to be semantically secure, we need joint circular security for the above two schemes. This assumption seems reasonable since the former building block scheme is also based on $\mathsf{MKHE}_{\mathsf{BHP}}$ (albeit an additional LWE-based OT protocol from [9]).

5 Maliciously circuit-private MPC based on LWE

In this section, we describe our MPC protocol from any maliciously circuit-private MKHE scheme with distributed setup, a maliciously (statistically) sender-private OT protocol, and a projective circuit garbling scheme.

5.1 Definitions for MPC

We here give the definition of maliciously circuit-private MPC protocols. We consider a 1-server and N-client protocol. In this protocol, inputs to participating parties are asymmetric (i.e., the server has a function to be computed, and the clients have inputs for the function), so the secure MPC protocol needs to have two types of security requirements: server privacy and client privacy. The server privacy has almost the same requirement with the circuit privacy in MKHE, and the client privacy means the semantic security of the client's input. We do not allow adversaries to corrupt the server, but we want to consider stronger security against clients. The client privacy should be considered for a semi-honest server and malicious clients.

In the following, we first define a client-server MPC protocol, then we define the client privacy and server privacy for the protocol, where we refer to the server privacy as circuit privacy. These definitions follow from those in [17].

Definition 5.1 *(Client-Server MPC protocol)* Let $\mathcal{C}$ be a class of functions with at most N inputs. A multi-party protocol Π for $\mathcal{C}$ is a protocol between parties $P_1, \ldots, P_N$, and the server S where P_i $(i = 1, \ldots, N)$ is given x_i as input, and S is given a function F on N inputs. Denote $\mathbf{x} = (x_1, \ldots, x_N)$. At the end of the protocol, each party P_i outputs $F(\mathbf{x})$ while S outputs $\perp$.

Definition 5.2 *(Client Privacy)* An adversary $\mathcal{A}$ corrupting $\{P_i\}_{i \in T}$ for some $T \subseteq [N]$ receives all messages directed to P_i for every $i \in T$ and controls the messages that they send. Let $\mathsf{View}_{\Pi,S}(F, \mathbf{x})$ denote the joint collection of messages that S receives in an execution of the protocol Π on an N-input function $F \in \mathcal{C}$ and an input set $\mathbf{x} = (x_1, \ldots, x_N)$. Let $\mathsf{View}_{\Pi,\mathcal{A}}(F, \mathbf{x})$ denote the joint collection of messages $\mathcal{A}$ receives through corrupted parties in an execution of protocol Π on F and $\mathbf{x}$. A multi-party protocol Π for $\mathcal{C}$ is secure against a semi-honest server and malicious clients if the protocol satisfies the following two privacy notions:

- **Privacy against clients** For every adversary $\mathcal{A}$ corrupting parties $\{P_i\}_{i \in T}$ with $|T| = t < N$, for all N-input functions $F \in \mathcal{C}$ and for all input sets $\mathbf{x} = (x_1, \ldots, x_N)$ and $\mathbf{x}' = (x'_1, \ldots, x'_N)$ such that $x_i = x'_i$ for any $i \in T$, for all y the range of F,

$$[\mathsf{View}_{\Pi,\mathcal{A}}(F, \mathbf{x}) : y = F(\mathbf{x})] \approx_c [\mathsf{View}_{\Pi,\mathcal{A}}(F, \mathbf{x}') : y = F(\mathbf{x}')].$$

- **Privacy against a server** For every server S, for all N-input functions $F \in \mathcal{C}$ and for all input sets $\mathbf{x} = (x_1, \ldots, x_N)$ and $\mathbf{x}' = (x'_1, \ldots, x'_N)$ such that $x_i = x'_i$ for any $i \in T$, for all y the range of F,

$$[\mathsf{View}_{\Pi,S}(F, \mathbf{x}) : y = F(\mathbf{x})] \approx_c [\mathsf{View}_{\Pi,S}(F, \mathbf{x}') : y = F((\mathbf{x}')].$$

Definition 5.3 *(Malicious Circuit Privacy (Server Privacy))* Let Π be a multi-party protocol for a function class $\mathcal{C}$. In the ideal-world execution of Π, the computation of $F \in \mathcal{C}$ is performed through a trusted functionality $\mathcal{F}$. Each party P_i sends their input x_i to $\mathcal{F}$, the server sends F to $\mathcal{F}$, which performs the computation and sends the output $F(\mathbf{x})$ to each P_i $(i \in [N])$. Let $\mathsf{Ideal}_{\mathcal{F},\mathcal{S}}(F, \mathbf{x})$ denote the joint output of the ideal-world adversary (called simulator) $\mathcal{S}$, parties $P_1, \ldots, P_N$ and the server S. Moreover, let $\mathsf{Real}_{\Pi,\mathcal{A}}(F, \mathbf{x})$ denote the joint output of the real-world adversary $\mathcal{A}$, parties $P_1, \ldots, P_N$ and the server S. The protocol Π is *maliciously circuit-private* if for every malicious (and possibly unbounded) adversary $\mathcal{A}$ corrupting any number of parties, there exists an unbounded simulator $\mathcal{S}$ with black-box access to $\mathcal{A}$ such that for all N-input functions $F \in \mathcal{C}$ and for all input sets $\mathbf{x} = (x_1, \ldots, x_N)$, $\mathsf{Ideal}_{\mathcal{F},\mathcal{S}}(F, \mathbf{x}) \approx_s \mathsf{Real}_{\Pi,\mathcal{A}}(F, \mathbf{x})$.

5.2 Construction

Our MPC protocol consists of four rounds, and is maliciously circuit-private in the plain model. We follow the approach of [17] but require an additional round (the first round) that computes the distributed setup.

Intuition of our construction The server evaluates the function over encrypted inputs sent from the clients. The server then constructs a special decryption circuit that given as input a secret key for the circuit-private MKFHE scheme and randomness used to generate the secret key, decrypts the result of the homomorphic evaluation if the input secret keys are valid. Also, the server sends a garbled circuit of the special decryption circuit to the clients. The difference between real-world and ideal-world executions of the protocol is how to construct a garbled circuit of the special decryption circuit. The simulator of the malicious circuit privacy for

our MPC protocol generates a simulated result of the homomorphic evaluation by using the simulator for the circuit-private homomorphic scheme, and compute a garbled circuit for the special decryption circuit hardwired with the simulated result. Since the simulated result is statistically indistinguishable from the result of the real evaluation, the garbled circuit is also statistically indistinguishable from the one in the real-world execution of the protocol.

Building blocks We will use the following building blocks.

- Let $\mathsf{MKHE}_{\mathsf{mCP}} = (\mathsf{dSetup}_{\mathsf{mCP}}, \mathsf{KG}_{\mathsf{mCP}}, \mathsf{Enc}_{\mathsf{mCP}}, \mathsf{Eval}_{\mathsf{mCP}}, \mathsf{Dec}_{\mathsf{mCP}})$ be a maliciously circuit-private MKFHE scheme with distributed setup.
- Let $\mathsf{OT} = (\mathsf{Q}, \mathsf{A}, \mathsf{D})$ be a maliciously sender-private OT protocol.
- Let $\mathsf{GC} = (\mathsf{GCircuit}, \mathsf{GEval})$ be a projective circuit garbling scheme.

The description of our protocol Π is as follows.

- **Inputs and outputs:** The clients P_i $(i = 1, ., N)$ is given x_i as input, and the server S is given a function F on N inputs. Each P_i outputs $F(x_1, ., x_N)$ while S outputs $\perp$.
- **Round 1:** For $i \in [N]$, the client P_i computes $\mathsf{pp}_i \xleftarrow{\$} \mathsf{dSetup}_{\mathsf{mCP}}(1^\lambda, 1^N, i \in [N])$, and broadcasts pp_i to other parties and the server S. Each of the parties P_i and the server S checks if $\mathsf{pp}_j \in [\mathsf{dSetup}_{\mathsf{mCP}}(1^\lambda, 1^N, j)$ for all $j \in [N]$. If the check fails, the parties abort the protocol. Let $\mathbf{pp} := (\mathsf{pp}_j)_{j\in[N]}$.
- **Round 2:** For $i \in [N]$, the client P_i runs $(\mathsf{pk}_i, \mathsf{sk}_i) \xleftarrow{\$} \mathsf{KG}_{\mathsf{mCP}}(\mathbf{pp}, i; r_{\mathsf{KG},i})$. Let s and r be the bit-size of sk_i and $r_{\mathsf{KG},i}$, respectively. The client P_i computes $(q_i^k, \mathsf{st}_i^k) \xleftarrow{\$} \mathsf{Q}(1^\lambda, \mathsf{sk}_{i,k})$ for $k \in [s]$ (where $\mathsf{sk}_{i,k}$ is the k-th bit of sk_i), and $(q_i^{s+k}, \mathsf{st}_i^{s+k}) \xleftarrow{\$} \mathsf{Q}(1^\lambda, r_{\mathsf{KG},i,k})$ for $k \in [r]$ (where $r_{\mathsf{KG},i,k}$ is the k-th bit of $r_{\mathsf{KG},i}$). Also P_i computes a ciphertext $\mathsf{ct}_i \xleftarrow{\$} \mathsf{Enc}_{\mathsf{mCP}}(\mathsf{pk}_i, x_i)$, and sends $(\mathsf{pk}_i, \mathsf{ct}_i, (q_i^k)_{k\in[s+r]})$ to the server S.
- **Round 3:** The input of the server S is a function F that takes on inputs $\mathbf{x} = (x_1, \ldots, x_N)$. The server S selects a circuit C representing the function F. It then computes $\widehat{\mathsf{ct}} \xleftarrow{\$} \mathsf{Eval}_{\mathsf{mCP}}(C, \mathbf{pp}, \mathbf{pk}, (\mathsf{ct}_j)_{j\in[N]})$, where we denote $\mathbf{pk} := (\mathsf{pk}_j)_{j\in[N]}$. Let $g_{\widehat{\mathsf{ct}},\mathbf{pp},\mathbf{pk}}$ be the following circuit:

$$g_{\widehat{\mathsf{ct}},\mathbf{pp},\mathbf{pk}}((\mathsf{sk}_j, r_j)_{j\in[N]}) := \begin{cases} \mathsf{Dec}(\mathbf{sk}, \widehat{\mathsf{ct}}) & \text{if } (\mathsf{pk}_j, \mathsf{sk}_j) = \mathsf{KG}_{\mathsf{mCP}}(\mathbf{pp}, j; r_j) \text{ for all } j \in [N]; \\ \perp & \text{otherwise} \end{cases},$$

  where we denote $\mathbf{sk} := (\mathsf{sk}_j)_{j\in[N]}$. The server S then generates a garbled circuit $(G, e) \xleftarrow{\$} \mathsf{GCircuit}(1^\lambda, g_{\widehat{\mathsf{ct}},\mathbf{pp},\mathbf{pk}})$, where $e = (X_j^0, X_j^1)_{j\in[N(r+s)]}$. For each $i \in [N]$ and $k \in [r + s]$, the server S computes $a_i^k \xleftarrow{\$} \mathsf{A}(q_i^k, X_{(i-1)(r+s)+k}^0, X_{(i-1)(r+s)+k}^1)$. It finally sends $(G, (a_i^k)_{k\in[r+s]})$ to the client P_i for each $i \in [N]$.
- **Round 4:** For $i \in [N]$, P_i computes the garbled input $X_{(i-1)(r+s)+k} = \mathsf{D}(a_i^k, \mathsf{st}_i^k)$ for $k \in [s + r]$. The client P_i broadcasts these to all the other clients, P_j for $j \in [N]\backslash\{i\}$. Each client computes $y := \mathsf{GEval}(G, (X_j)_{j\in[N(r+s)]})$.

The security of our protocol is guaranteed by the following theorem.

Theorem 5.1 *If $\mathsf{MKHE}_{\mathsf{mCP}}$ is semantically secure and maliciously circuit-private, OT is statistically sender private against malicious receivers, and GC is secure, then Π is a secure MPC protocol with malicious circuit privacy.*

The proof of Theorem 5.1 is divided into the following three lemmas: the first two lemmas state that the protocol Π has privacy against malicious clients and a semi-honest server, and the third one is for saying the malicious circuit privacy of Π.

Lemma 5.1 (Privacy against Malicious Clients) *Assume $\mathsf{MKHE}_{\mathsf{mCP}}$ is semantically secure, OT is statistically sender-private against malicious receivers and GC is secure. Then, for any $y \in \{0,1\}$, $\mathbf{x}, \mathbf{x}' \in \{0,1\}^N$, $F : \{0,1\}^N \to \{0,1\}$ and $T \subset [N]$ such that $x_i = x'_i$ for any $i \in T$,*

$$[\mathit{View}_{\Pi,\mathcal{A}}(F, \mathbf{x}) : y = F(\mathbf{x})] \approx_c [\mathit{View}_{\Pi,\mathcal{A}}(F, \mathbf{x}') : y = F(\mathbf{x}')].$$

Proof The ciphertexts $\{\mathsf{ct}_j\}_{j\in[N]\setminus T}$ obtained for $\mathbf{x}$ in the second round of the protocol are computationally indistinguishable from $\{\mathsf{ct}'_j\}_{j\in[N]\setminus T}$ obtained for $\mathbf{x}'$ by the semantic security of $\mathsf{MKHE}_{\mathsf{mCP}}$. Let g and g' be circuits obtained in the third round of Π for input $\mathbf{x}$ and $\mathbf{x}'$, respectively, where both circuits evaluate to y if a secret key and randomness for the key generation are valid, and 0 otherwise. Let $(G, e) \xleftarrow{\$} \mathsf{GCircuit}(1^\lambda, g)$ and $(G', e') \xleftarrow{\$} \mathsf{GCircuit}(1^\lambda, g')$. By the sender-privacy of OT against malicious receivers, $\mathcal{A}$ can obtain at most one garbled input (token) for the corrupted clients. Let X and X' be garbled inputs obtained by e and e' for the secret key and its randomness. Since both circuits evaluate to y on garbled inputs corresponding to valid secret keys and 0 otherwise we have $(G, X) \approx_c (G', X')$ by the security of GC. Also, a_i's and garbled inputs do not depend on the protocol inputs. Therefore, the views of an adversary $\mathcal{A}$ in the executions of Π for input $\mathbf{x}$ and that for $\mathbf{x}'$ are computationally indistinguishable. This concludes the proof of lemma. □

Lemma 5.2 (Privacy against a Semi-honest Server) *If $\mathsf{MKHE}_{\mathsf{mCP}}$ is semantically secure and OT is receiver private against semi-honest senders, then for any $y \in \{0,1\}$, $\mathbf{x}, \mathbf{x}' \in \{0,1\}^N$, and $F : \{0,1\}^N \to \{0,1\}$, we have*

$$[\mathit{View}_{\Pi,S}(F, \mathbf{x}) : y = F(\mathbf{x})] \approx_c [\mathit{View}_{\Pi,S}(F, \mathbf{x}') : y = F(\mathbf{x}')].$$

Proof In the second round of Π, each client P_i $(i = 1, \ldots, N)$ sends OT-queries $(q_i^k)_{k\in[r+s]}$ to the server, and the queries do not depend on the protocol input. Thus, by the receiver privacy of OT, they are computationally indistinguishable from OT-queries generated by using some fixed string independent of the secret key and randomness. This in turn enables us to invoke the semantic security of $\mathsf{MKHE}_{\mathsf{mCP}}$, and say that the server's view in the case that the ciphertext ct_i in the second round message encrypts $\mathbf{x}$, is computationally indistinguishable from the case that ct_i encrypts $\mathbf{x}'$. This concludes the proof of the lemma. □

Intuition for the proof of the following theorem is given in the beginning of this section (Sect. 5.2).

Lemma 5.3 (Malicious Circuit Privacy) *If $\mathsf{MKHE}_{\mathsf{mCP}}$ is maliciously circuit-private, then the protocol Π is maliciously circuit-private.*

Proof Let $\mathsf{Ext}_{\mathsf{mCP}}$ and $\mathsf{Sim}_{\mathsf{mCP}}$ be the extractor and simulator that are guaranteed to exist by the malicious circuit privacy of $\mathsf{MKHE}_{\mathsf{mCP}}$. Here we show the construction of a simulator $\mathcal{S}$ for Π, where we can assume w.l.o.g. that the real-world adversary $\mathcal{A}$ corrupts all the clients, because we are not considering the privacy of the clients here.

- **Step 1**: The simulator $\mathcal{S}$ receives pp_j^* for $j \in T = [N]$ from $\mathcal{A}$. Denote $\mathbf{pp}^* := (\mathsf{pp}_j^*)_{j\in[N]}$.

- **Step 2**: Given $(\mathsf{pk}_i^*, \mathsf{ct}_i^*, (q_{i,k}^*)_{k\in[r+s]})_{i\in T}$ from $\mathcal{A}$, $\mathcal{S}$ runs $\mathsf{Ext}_{\mathsf{mCP}}$ to compute the corrupted input $\tilde{x_i} := \mathsf{Ext}_{\mathsf{mCP}}(\mathbf{pp}^*, i, \mathsf{pk}_i^*, \mathsf{ct}_i^*)$. It then submits them to the ideal functionality $\mathcal{F}$ and obtains $b^* := F(\tilde{x}_1, \ldots, \tilde{x}_N)$.
- **Step 3**: Denote $\mathbf{pk}^* = (\mathsf{pk}_j^*)_{j\in[N]}$. The simulator $\mathcal{S}$ runs $\mathsf{Sim}_{\mathsf{mCP}}$ to compute

$$\widehat{\mathsf{ct}}^* \xleftarrow{\$} \mathsf{Sim}_{\mathsf{mCP}}(\mathbf{pp}^*, \mathbf{pk}^*, (j, \mathsf{ct}_j^*)_{j\in[N]}, b^*),$$

then defines a circuit

$$g_{\widehat{\mathsf{ct}}^*,\mathbf{pp}^*,\mathbf{pk}^*}((\mathsf{sk}_j, r_j)_{j\in[N]}) := \begin{cases} \mathsf{Dec}_{\mathsf{mCP}}(\mathsf{sk}_1, \ldots, \mathsf{sk}_N, \widehat{\mathsf{ct}}^*) & \text{if } (\mathsf{pk}_j^*, \mathsf{sk}_j) = \mathsf{KG}_{\mathsf{mCP}}(\mathbf{pp}^*, j; r_j) \\ & \quad \text{for all } j \in [N] \\ 0 & \text{otherwise} \end{cases}.$$

The simulator generates a garbled circuit $(G^*, e^*) \xleftarrow{\$} \mathsf{GCircuit}(1^\lambda, g_{\widehat{\mathsf{ct}}^*,\mathbf{pp}^*,\mathbf{pk}^*})$ where $e^* = (X_j^{*0}, X_j^{*1})_{j\in[N(r+s)]}$. For all $i \in [N]$ and $k \in [r+s]$, the simulator $\mathcal{S}$ computes $a_{i,k}^* \xleftarrow{\$} \mathsf{A}(q_i^{*k}, X_{(i-1)(r+s)+k}^{*0}, X_{(i-1)(r+s)+k}^{*1})$, and sends $(G^*, (a_{i,k}^*)_{k\in[r+s]})$ for $i \in T$ to $\mathcal{A}$.
- **Output**: In Round 4 of the protocol, no interaction between the server S and the clients P_i occurs. Since we are assuming that all clients are corrupted and controlled by $\mathcal{A}$, the simulator $\mathcal{S}$ need not do any action until $\mathcal{A}$ returns the outputs of the corrupted parties. Finally, when $\mathcal{A}$ terminates with the outputs of the corrupted parties in the real-world execution of Π, $\mathcal{S}$ just forwards the received outputs and terminates.

By the malicious circuit privacy of $\mathsf{MKHE}_{\mathsf{mCP}}$, $\widehat{\mathsf{ct}}^*$ is statistically indistinguishable from the real computation of $\mathsf{Eval}_{\mathsf{mCP}}$ for F, and thus the garbled circuit G^* is also statistically indistinguishable from the one in the real-world execution of Π. Hence we have $\mathsf{Real}_{\Pi,\mathcal{A}}(F, \mathbf{x}) \approx_s \mathsf{Ideal}_{\Pi,\mathcal{S}}(F, \mathbf{x})$. □

Instantiations from LWE When instantiated $\mathsf{MKHE}_{\mathsf{mCP}}$ with our proposed schemes in Sects. 4.2 and 4.3, we obtain Theorems 1.3 and 1.4, respectively.

6 Conclusion

In this paper, we constructed a maliciously circuit-private MKHE scheme for branching programs based on LWE as well as weak circular security of the (plain) MKFHE scheme by Brakerski et al. [15]. We also showed how the obtained scheme can be combined with a plain MKFHE scheme (such as [15]), to obtain a fully homomorphic maliciously circuit-private scheme. To obtain this result, we relied on a somewhat non-standard circular security assumption, but a similar assumption is required to show the security of the previous maliciously circuit-private MKFHE scheme in [17]. Based on either of our MK(F)HE schemes, we built a four-round MPC protocol with circuit privacy against semi-honest server and malicious clients in the plain model, where the additional building blocks required for our protocol can be instantiated assuming only LWE.

Funding Nuttapong Attrapadung, Goichiro Hanaoka, Takahiro Matsuda, and Jacob C. N. Schuldt were partly supported by JSPS KAKENHI Kiban-A Grant Number 19H01109.

Availability of data and materials Not applicable.

Declarations

Conflict of interest/Competing interests Not applicable.

Code availability Not applicable.

References

1. Albrecht M.R., Bai S., Ducas L.: A subfield lattice attack on overstretched NTRU assumptions - cryptanalysis of some FHE and graded encoding schemes. In: Robshaw M., Katz J. (eds.) CRYPTO 2016, Part I, volume 9814 of LNCS, pp. 153–178, Santa Barbara, CA, USA, August 14–18 (2016). Springer, Heidelberg, Germany. https://doi.org/10.1007/978-3-662-53018-4_6.
2. Ananth P., Jain A., Jin Z., Malavolta G.: Multi-key fully-homomorphic encryption in the plain model. In TCC 2020, (2020).
3. Badrinarayanan S., Jain A., Manohar N., Sahai A.: Threshold multi-key FHE and applications to round-optimal MPC. Cryptology ePrint Archive, Report 2018/580 (2018). https://eprint.iacr.org/2018/580/20190529:195715.
4. Barrington D.A. Mix.: Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. In: 18th ACM STOC, pp. 1–5, Berkeley, CA, USA, May 28–30, (1986). ACM Press. https://doi.org/10.1145/12130.12131.
5. Beaver D., Micali S., Rogaway P.: The round complexity of secure protocols (extended abstract). In: 22nd ACM STOC, pp. 503–513, Baltimore, MD, USA, May 14–16, (1990). ACM Press. https://doi.org/10.1145/100216.100287.
6. Ben-Or M., Goldwasser S., Wigderson A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In: 20th ACM STOC, pp. 1–10, Chicago, IL, USA, May 2–4 (1988). ACM Press. https://doi.org/10.1145/62212.62213.
7. Bendlin R., Damgård I., Orlandi C., Zakarias S.: Semi-homomorphic encryption and multiparty computation. In: Paterson K.G. (Ed.) EUROCRYPT 2011, volume 6632 of LNCS, pp. 169–188, Tallinn, Estonia, May 15–19 (2011). Springer, Heidelberg. https://doi.org/10.1007/978-3-642-20465-4_11.
8. Brakerski Z.: Fully homomorphic encryption without modulus switching from classical GapSVP. In: Reihaneh S.-N., Ran C. (eds.) CRYPTO 2012, volume 7417 of LNCS, pp. 868–886, Santa Barbara, CA, USA, August 19–23 (2012). Springer, Heidelberg. https://doi.org/10.1007/978-3-642-32009-5_50.
9. Brakerski Z., Döttling, N.: Two-message statistically sender-private OT from LWE. In: Beimel A., Dziembowski S. (eds.) TCC 2018, Part II, volume 11240 of LNCS, pages 370–390, Panaji, India, November 11–14 (2018). Springer, Heidelberg, Germany. https://doi.org/10.1007/978-3-030-03810-6_14.
10. Brakerski Z., Perlman R.: Lattice-based fully dynamic multi-key FHE with short ciphertexts. In: Robshaw M., Katz J. (eds.) CRYPTO 2016, Part I, volume 9814 of LNCS, pp. 190–213, Santa Barbara, CA, USA, August 14–18 (2016). Springer, Heidelberg, Germany. https://doi.org/10.1007/978-3-662-53018-4_8.
11. Brakerski Z., Vaikuntanathan V.: Fully homomorphic encryption from ring-LWE and security for key dependent messages. In: Rogaway P. (ed.) CRYPTO 2011, volume 6841 of LNCS, pp. 505–524, Santa Barbara, CA, USA, August 14–18 (2011a). Springer, Heidelberg. https://doi.org/10.1007/978-3-642-22792-9_29.
12. Brakerski Z., Vaikuntanathan V.: Efficient fully homomorphic encryption from (standard) LWE. In: Ostrovsky R. (ed.) 52nd FOCS, pp. 97–106, Palm Springs, CA, USA, October 22–25 (2011b). IEEE Computer Society Press. https://doi.org/10.1109/FOCS.2011.12.

13. Brakerski Z., Gentry C., Vaikuntanathan V.: (Leveled) fully homomorphic encryption without bootstrapping. In Goldwasser S (ed.) ITCS 2012, pp. 309–325, Cambridge, MA, USA, January 8–10 (2012). ACM. https://doi.org/10.1145/2090236.2090262.
14. Brakerski Z., Vaikuntanathan V., Wee H., Wichs D.: Obfuscating conjunctions under entropic ring LWE. In Sudan M (ed.) ITCS 2016, pp. 147–156, Cambridge, MA, USA, January 14–16 (2016). ACM. https://doi.org/10.1145/2840728.2840764.
15. Brakerski Z., Halevi S., Polychroniadou A.: Four round secure computation without setup. In: Kalai Y., Reyzin L. (eds.) TCC 2017, Part I, volume 10677 of LNCS, pp. 645–677, Baltimore, MD, USA, November 12–15 (2017). Springer, Heidelberg, Germany. https://doi.org/10.1007/978-3-319-70500-2_22.
16. Chongchitmate W., Ostrovsky R.: Circuit-private multi-key FHE. Cryptology ePrint Archive, Report 2017/010 (2017a). http://eprint.iacr.org/2017/010.
17. Chongchitmate W., Ostrovsky R.: Circuit-private multi-key FHE. In Fehr S. (ed.) PKC 2017, Part II, volume 10175 of LNCS, pp. 241–270, Amsterdam, The Netherlands, March 28–31 (2017b). Springer, Heidelberg. https://doi.org/10.1007/978-3-662-54388-7_9.
18. Clear M., McGoldrick C.: Multi-identity and multi-key leveled FHE from learning with errors. In Gennaro R., Robshaw M.J.B. (eds.) CRYPTO 2015, Part II, volume 9216 of LNCS, pp. 630–656, Santa Barbara, CA, USA, August 16–20 (2015). Springer, Heidelberg, Germany. https://doi.org/10.1007/978-3-662-48000-7_31.
19. Damgård I., Pastro V., Smart N.P., Zakarias S.: Multiparty computation from somewhat homomorphic encryption. In: Safavi-Naini R., Canetti R. (eds.) CRYPTO 2012, volume 7417 of LNCS, pp. 643–662, Santa Barbara, CA, USA, August 19–23 (2012). Springer, Heidelberg, Germany. https://doi.org/10.1007/978-3-642-32009-5_38.
20. Demmler D., Schneider T., Zohner M.: ABY—A framework for efficient mixed-protocol secure two-party computation. In NDSS 2015, San Diego, CA, USA, February 8–11 (2015). The Internet Society.
21. Gentry C.: A Fully Homomorphic Encryption Scheme. PhD thesis, Stanford University, Available at http://crypto.stanford.edu/craig (2009a).
22. Gentry C.: Fully homomorphic encryption using ideal lattices. In Mitzenmacher M. (ed.) 41st ACM STOC, pp. 169–178, Bethesda, MD, USA, May 31–June 2 (2009b). ACM Press. https://doi.org/10.1145/1536414.1536440.
23. Gentry C., Sahai A., Waters B.: Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In: Canetti R., Garay J.A. (eds) CRYPTO 2013, Part I, volume 8042 of LNCS, pages 75–92, Santa Barbara, CA, USA, August 18–22 (2013). Springer, Heidelberg, Germany. https://doi.org/10.1007/978-3-642-40041-4_5.
24. Goldreich O., Micali S., Wigderson A.: How to play any mental game or A completeness theorem for protocols with honest majority. In: Aho A (ed.) 19th ACM STOC, pp. 218–229, New York City, NY, USA, May 25–27 (1987). ACM Press. https://doi.org/10.1145/28395.28420.
25. Goldwasser S., Kalai Y.T., Peikert C., Vaikuntanathan V.: Robustness of the learning with errors assumption. In Yao A.C.-C. (ed.) ICS 2010, pp. 230–240, Tsinghua University, Beijing, China, January 5–7 (2010). Tsinghua University Press.
26. Hazay C., Scholl P., Soria-Vazquez E.: Low cost constant round MPC combining BMR and oblivious transfer. In Takagi T., Peyrin T. (eds.) ASIACRYPT 2017, Part I, volume 10624 of LNCS, pages 598–628, Hong Kong, China, December 3–7 (2017). Springer, Heidelberg, Germany. https://doi.org/10.1007/978-3-319-70694-8_21.
27. Ishai Y., Kushilevitz E.: Perfect constant-round secure computation via perfect randomizing polynomials. In: Widmayer P., Ruiz F.T., Bueno R.M., Hennessy M., Eidenbenz S., Conejo R. (eds.) editors, ICALP 2002, volume 2380 of LNCS, pp. 244–256, Malaga, Spain, July 8–13 (2002). Springer, Heidelberg, Germany. https://doi.org/10.1007/3-540-45465-9_22.
28. Ishai Y., Paskin A.: Evaluating branching programs on encrypted data. In Vadhan S.P. (ed.) TCC 2007, volume 4392 of LNCS, pp. 575–594, Amsterdam, The Netherlands, February 21–24 (2007). Springer, Heidelberg, Germany. https://doi.org/10.1007/978-3-540-70936-7_31.
29. Kamara S.., Mohassel P., Raykova M.: Outsourcing multi-party computation. Cryptology ePrint Archive, Report 2011/272 (2011). http://eprint.iacr.org/2011/272.
30. Lindell Y., Smart N.P., Soria-Vazquez E.: More efficient constant-round multi-party computation from BMR and SHE. In: Hirt M., Smith A.D. (eds.) TCC 2016-B, Part I, volume 9985 of LNCS, pages 554–581, Beijing, China, October 31 – November 3 (2016). Springer, Heidelberg, Germany. https://doi.org/10.1007/978-3-662-53641-4_21.
31. López-Alt A., Tromer E., Vaikuntanathan V.: On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In Karloff H.J., Pitassi T. (eds.) 44th ACM STOC, pages 1219–1234, New York, NY, USA, May 19–22 (2012). ACM Press. https://doi.org/10.1145/2213977.2214086.

32. Mohassel P., Rindal P.: ABY^3: A mixed protocol framework for machine learning. In: Lie D., Mannan M., Backes M., Wang X.F. (eds.) ACM CCS 2018, pp. 35–52, Toronto, ON, Canada, October 15–19 (2018). ACM Press. https://doi.org/10.1145/3243734.3243760.
33. Mohassel P., Sadeghian S.S.: How to hide circuits in MPC an efficient framework for private function evaluation. In Johansson T., Nguyen P.Q. (eds.) EUROCRYPT 2013, volume 7881 of LNCS, pp. 557–574, Athens, Greece, May 26–30 (2013). Springer, Heidelberg, Germany. https://doi.org/10.1007/978-3-642-38348-9_33.
34. Mohassel P., Sadeghian S.S., Smart N.P.: Actively secure private function evaluation. In Sarkar P., Iwata T. (eds.) ASIACRYPT 2014, Part II, volume 8874 of LNCS, pp. 486–505, Kaoshiung, Taiwan, R.O.C., December 7–11 (2014). Springer, Heidelberg, Germany. https://doi.org/10.1007/978-3-662-45608-8_26.
35. Mukherjee P., Wichs D.: Two round multiparty computation via multi-key FHE. In: Fischlin M., Coron J.-S. (eds.) EUROCRYPT 2016, Part II, volume 9666 of LNCS, pp. 735–763, Vienna, Austria, May 8–12 (2016). Springer, Heidelberg, Germany. https://doi.org/10.1007/978-3-662-49896-5_26.
36. Ostrovsky R., Paskin-Cherniavsky A., Paskin-Cherniavsky B.: Maliciously circuit-private FHE. In Juan A. Garay and Rosario Gennaro, editors, CRYPTO 2014, Part I, volume 8616 of LNCS, pp. 536–553, Santa Barbara, CA, USA, August 17–21 (2014). Springer, Heidelberg, Germany. https://doi.org/10.1007/978-3-662-44371-2_30.
37. Raz R.: Elusive functions and lower bounds for arithmetic circuits. In Ladner R.E., Dwork C. (eds.) 40th ACM STOC, pp. 711–720, Victoria, BC, Canada, May 17–20 (2008). ACM Press. https://doi.org/10.1145/1374376.1374479.
38. Regev O.: On lattices, learning with errors, random linear codes, and cryptography. In: Gabow H.N., Fagin R. (eds.) 37th ACM STOC, pp. 84–93, Baltimore, MA, USA, May 22–24 (2005). ACM Press. https://doi.org/10.1145/1060590.1060603.
39. Valiant L.G.: Universal circuits (preliminary report). In Chandra A.K., Wotschke D., Friedman E.P., Harrisonl M.A. (eds.) 8th ACM STOC, pp. 196–203, New York, NY, USA, May 3–5 (1976). https://doi.org/10.1145/800113.803649.
40. van Dijk M., Gentry C., Halevi S., Vaikuntanathan V.: Fully homomorphic encryption over the integers. In Gilbert H (ed.) EUROCRYPT 2010, volume 6110 of LNCS, pp. 24–43, French Riviera, May 30–June 3 (2010). Springer, Heidelberg, Germany. https://doi.org/10.1007/978-3-642-13190-5_2.
41. Yao A.C.-C.: How to generate and exchange secrets (extended abstract). In: 27th FOCS, pp. 162–167, Toronto, Ontario, Canada, October 27–29 (1986). IEEE Computer Society Press. https://doi.org/10.1109/SFCS.1986.25.