mathematics

MDPI

Article

An Examination of Multi-Key Fully Homomorphic Encryption and Its Applications

Minghao Yuan [1], Dongdong Wang [2], Feng Zhang [3], Shenqing Wang [3], Shan Ji [3,*] and Yongjun Ren [1]

[1] Engineering Research Center of Digital Forensics, Ministry of Education, School of Computer Science, Nanjing University of Information Science and Technology, Nanjing 210044, China
[2] The 15th Research Institute of China Electronics Technology Group Corporation, Beijing 100083, China
[3] College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China
* Correspondence: shanji@nuaa.edu.cn

Abstract: With the rapid development of the Internet of Things (IoT) technology, the security problems it faces are increasingly prominent and have attracted much attention in industry and the academy. Traditional IoT architecture comes with security risks. Illegal intrusion of attackers into the network layer disrupts the availability of data. The untrusted transmission environment increases the difficulty of users sharing private data, and various outsourced computing and application requirements bring the risk of privacy leakage. Multi-key fully homomorphic encryption (MKFHE) realizes operations between ciphertexts under different key encryption and has great application potential. Since 2012, the first MKFHE scheme LTV12 has been extended from fully homomorphic encryption (FHE) and has ignited the enthusiasm of many cryptographic researchers due to its lattice-based security and quantum-resistant properties. According to its corresponding FHE scheme, the MKFHE schemes can be divided into four kinds: Gentry–Sahai–Water (GSW), number theory research unit (NTRU), Brakerski–Gentry–Vaikuntanathan (BGV), and FHE over the tour (TFHE). Efficiency and cost are urgent issues for MKFHE. New schemes are mainly improved versions of existing schemes. The improvements are mostly related to the four parts of MKFHE: security assumption, key generation, plaintext encryption, and ciphertext processing. We classified MKFHE schemes according to the improved partial schemes, and we present some improved techniques and the applications of MKFHE.

Keywords: multi-key fully homomorphic encryption; Internet of Things; distributed computing; privacy protection

MSC: 94A60; 06B99; 68P27

check for updates

Citation: Yuan, M.; Wang, D.; Zhang, F.; Wang, S.; Ji, S.; Ren, Y. An Examination of Multi-Key Fully Homomorphic Encryption and Its Applications. *Mathematics* **2022**, *10*, 4678. https://doi.org/10.3390/math10244678

Academic Editors: Liehuang Zhu, Meng Li and Zijian Zhang

Received: 8 November 2022
Accepted: 7 December 2022
Published: 9 December 2022

1. Introduction

The Internet of Things (IoT) technology has been widely applied in various domains, such as the military, industry, logistics, medical care, and smart homes [1], in recent years. The era of the Internet of everything has arrived, and the interactions between hundreds of millions of terminal devices generate massive data. An important method to deal with massive data is to apply distributed processing model represented by cloud computing and federated learning, which can make full use of idle resources of IoT devices. The IoT involves most aspects of daily life. It inevitably needs to collect people's personal information (such as consumption habits, travel routes, etc.). How the processing layer protects the privacy of all parties involved in processing is an urgent privacy protection problem.

Data security requires not only the security of the stored data but also the security of data processing. Fully homomorphic encryption (FHE) [2,3] allows arbitrary operations to be performed on encrypted data, with the same effect on the ciphertext as on the plaintext. Therefore, the users only have to upload data that have been encrypted with the user's

public key to the cloud to ensure that the data are stored and processed securely [4]. For example, there are already many applications of homomorphic encryption in the IoT combined with cloud computing. Users store the encrypted data in the cloud, and the decryption of the encrypted data needs to be jointly authorized by the user's terminal and the cloud. However, the existing schemes basically use single-key homomorphic encryption, but similar scenarios such sa cloud computing and federated learning usually involve handling multi-user data, where different users jointly participate in the same operation process and the operation results should be jointly decrypted by the participating users. FHE [5] only supports the operation of ciphertext encrypted with the same key, and different users holding the same secret key obviously cannot meet the security requirements.

The properties of multi-key fully homomorphic encryption (MKFHE) can be perfectly applied to the multi-user model. Users can obtain different keys from the same key generation algorithm, and the ciphertexts encrypted under different keys can be operated arbitrarily. The decryption process needs to be jointly decrypted by each user, which can solve the security and privacy problems in distributed computing scenarios. Additionally, because MKFHE is constructed based on the lattice difficulty problem [6], it possesses the ability to resist quantum attacks [7], which meets the current demand for resisting quantum computer threats.

Efficiency and cost are the biggest problems between the current MKFHE and the applications. Once the issues of efficiency and cost are overcome, MKFHE will be the preferred choice for security protection in multi-user environments. To solve this problem, researchers reduce the sizes of public and private keys, reduce the size of ciphertext, design more efficient ciphertext calculation methods, and use batch processing and compression ciphertext to improve efficiency and reduce the cost of MKFHE. Not all researchers have focused on the study of the cost and efficiency of MKFHE. For example, some researchers noticed that the current MKFHE schemes are all under the CRS (commom reference string) setting and that the ability of each user to independently generate keys is limited, so they designed the first MKFHE under the non-CRS setting. Some researchers noticed the security risks of the difficult assumption problem and constructed a new scheme based on the optimized difficult assumption problem.

The optimization of MKFHE is mostly carried out in the stages of safety assumption, key generation, plaintext encryption, and ciphertext processing, with the objectives of reducing key size, improving encryption efficiency, creating more efficient extension methods, and making smaller extended ciphertexts. There are also optimizations of security assumptions that choose the non-traditional MKFHE hard problem.

In this article, we present a classification and introduction of MKFHE schemes, show some of the technologies and definitions of the optimized schemes, introduce the applications of MKFHE, and summarize the current developments in MKFHE.

Multi-Key Fully Homomorphic Encryption

Leveled MKFHE [8]: Given a safety parameter and operation circuit C with the depth of L, a leveled MKFHE scheme is a tuple of efficient randomized algorithms (**Setup**, **KeyGen**, **Enc**, **Extend**, **Eval**, **Dec**) described as follows:

Setup (1^{λ}, 1^{k}, 1^{L}): Given the security parameter λ, a bound K on the number of keys, and a bound L on the circuit depth, output a public parameter pp.

KeyGen (pp): Input the public parameter pp; output public key pk_i and secret key sk_i (i = 1, . . . , K) for each party, key pk_i for ciphertext extension, and key evk_i for homomorphic evaluation.

Enc (pk_i, m_i): Input the public key pk_i of party i and a message m_i output a ciphertext c_i which contains the relevant private key and circuit-level information.

Extend (c_i, $pk_s = \{pk_{i1}, \ldots, pk_{ik}\}$): Input ciphertext c_i and key ek_i for ciphertext; output extended ciphertext $\hat{c}_{i,S}$. The corresponding user set is S_i and the public key set corresponding to the user set S ($S_i \subset$ S) to be expanded is $pk_s = \{pk_i, \ldots, pk_{ik}\}$. The corresponding private key is composed or calculated by the

private keys of all users in S in a specific form. Notice that the ciphertext extension algorithm is not necessary for all MKFHE schemes (number theory research unit (NTRU) type MKFHE does not need to expand the ciphertext), not all the ciphertext extension process require extended key ek_i (the ciphertext extension process of Brakerski–Gentry–Vaikuntanathan (BGV) type MKFHE does not require ek_i).

Eval (pp, C, $(\hat{c}_{1,S}, evk_1), \ldots, (\hat{c}_{t,S}, evk_t)$): Input a boolean circuit C and the tuple $(\hat{c}_{1,S}, evk_i)_{i=[t]}$ corresponding to the same user set S (which can be implemented by ciphertext extension); output $\hat{c}_S$ after homomorphic evaluation.

Dec $(sk_S, \hat{c}_S)$: Input a ciphertext $\hat{c}_S$ corresponding to a set of parties S = $\{i_1, \ldots, i_k\} \subseteq$[K] and joint private key $sk_s = \{sk_{i1}, sk_{i2}, \ldots, sk_{ik}\}$, which is composed or calculated by the private keys of all participants in a specific form; output the message m_S.

Correctness: To a leveled MKFHE, input any circuit C of depth at most L having t input wires and any tuples $(c_{i,S}, evk_i)_{i\in[t]}$, while letting μ_i = **Dec** $(sk_S, c_{i.S})$; then the MKFHE scheme is correct if and only if the following formula holds:

$$Pr[\textbf{Dec}\ (sk_S, \textbf{Eval}\ (C, (c_i, S, evk_i)_{i\in[t]})) \neq C\ (\mu_1, \ldots, \mu_t)] = negl\ (\lambda) \quad (1)$$

Compactness: To a leveled MKFHE, if there exists a polynomial poly $(\cdot)$ such that $|\ c\ | \leq$ poly $(\lambda$, K, L), and the length of c is independent of the circuit C, then the MKFHE scheme is compact. In general, the ciphertext length of MKFHE scheme is related to security parameter λ, the number of participants K, and the polynomial level of circuit depth L.

The properties of homomorphic evaluation are presented in Figure 1.

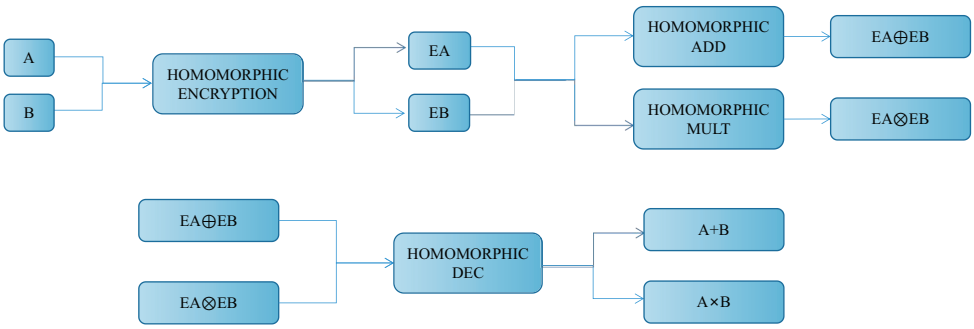

Figure 1. Homomorphic addition and multiplication.

Taking A and B as an example, the encrypted ciphertexts are EA and EB. After the homomorphic addition and the homomorphic multiplication of EA and EB, the decryption of the homomorphic operation has the same result as the direct addition and multiplication of A and B.

2. MKFHE Scheme Classification

2.1. MKFHE Classified by Improvement Steps

The biggest problems that stand between MKFHE schemes and their application are efficiency and cost. After the three classical types of MKFHE schemes were proposed, subsequent schemes were mostly constructed by improving on the various steps of the classical schemes. In this paper, MKFHE schemes are divided into four categories according to the different steps of improvement.

(a) The first category involves schemes that have improvements in security assumptions. Che et al. proposed the NTRU-type MKFHE scheme CZL20 [9] for prime cyclotomic rings in 2020, and other schemes that improve security assumptions.

(b) The second category involves schemes that have improvements in the key generation step. Kim et al. constructed the common random/reference string (CRS)-free KLP18 [10] scheme in 2018, which improves the MW16 [11] scheme, and some subsequently improved schemes.

(c) The third category involves schemes that have improvements in the encryption step. Chen et al. constructed the first CZW17 [12] scheme supporting multi-bit encryption based on BGV12 [13] in 2017, and subsequently improved schemes and multi-bit encryption schemes implemented in other ways.

(d) The fourth category involves schemes that have improvements in the ciphertext processing step. The CZW17 scheme implements an efficient ciphertext extension to obtain compact ciphertexts, and other schemes that improve on ciphertext extension, and the SWC21 [14] scheme that is able to compress ciphertexts.

The flowchart of the development of MKFHE according to this classification is shown in Figure 2.

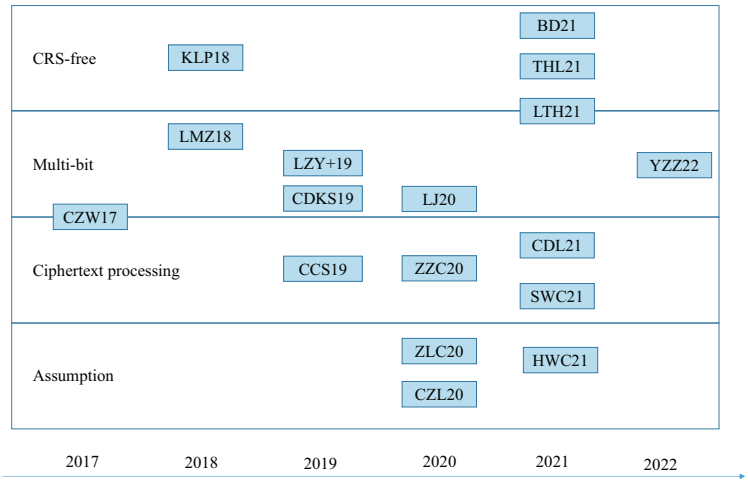

Figure 2. MKFHE developments by improvement steps.

2.2. Security Assumptions

The security of MKFHE schemes is based on their assumptions. Some of the assumptions used in the schemes have security challenges. Current NTRU-type MKFHE schemes are mainly constructed based on a polynomial ring of order 2 in powers; therefore, the security of these NTRU schemes may be threatened by subdomain attacks [15]. The security of the scheme can be enhanced by optimizing the security assumption problem.

The summary of Section 2.2 is shown in Table 1.

Table 1. Summary of Section 2.2.

Scheme	Technology	Function	Emphasis
CZL20	LBD and DEC	reduce the ciphertext size	cost and efficient
ZLC20	prime power cyclotomic polynomial ring	resist subdomain attacks	security
HWC21	learning-with-rounding (LWR)	avoid gaussion sampling	efficient and security

In 2020, Che et al. proposed an NTRU-type MKFHE scheme, CZL20 [9], based on the LTV12 [8]. The security of the scheme is based on the RLWE and DSPR assumptions over prime cyclotomic rings, and it has been proven that the NTRU encryption on prime partitioned rings has the potential to resist subdomain attacks. Che analyzed the impact elements of the LTV12 homomorphic evaluation process and proposed a low bit drop and dimensionally extended ciphertext (LBD&DEC) technique and a homomorphic multiplicative decryption structure for the NTRU that can eliminate the key-switching in the LTV12 scheme, and a modulus reduction technique is also applied to reduce ciphertext's dimension.

In 2020, Zhou et al. constructed the ZLC20 [16] scheme by replacing the power of two cyclotomic rings in the LTV12 scheme with a prime-power cyclotomic polynomial ring, which increased the number of optional ring structures during practical applications

and makes the scheme able to resist subdomain attacks. Additionally, the key generation algorithm was optimized by using a Gaussian distribution under regular embedding based on the NTRU scheme [17], which is also based on prime cyclotomic rings.

In 2021, Huang et al. constructed a fully dynamic multi-hop (users can join the operation at any time) Gentry–Sahai–Water (GSW)-type MKFHE scheme HWC21 [18]. Its security is based on the learning with rounding (LWR) problem [19], which has similar performance to the BP16 [20] scheme. The LWR assumption allows the scheme to avoid the time-consuming Gaussian sampling ("research has shown that Gaussian sampling may create side-channel vulnerabilities leading to key leakage") required in the learning with error (LWE) problem while sustaining almost the same security level.

2.3. Key Generation

Key generation is an important step in the actual application. Generally, when a single-key scheme is extended to a multi-key scheme, the public keys of different users are linked to each other by a CRS generated in the setup phase to enable ciphertexts under different key encryption to perform homomorphic operations. However, this weakens the ability of each user to generate public keys independently. Therefore, some researchers want to develop a MKFHE scheme without CRS in the public parameters.

The summary of Section 2.3 is shown in Table 2.

Table 2. Summary of Section 2.3.

Scheme	Technology	Function	Emphasis
KLP18	LinkAlgo	convert single key to multi-key scheme	decentralization
LTH21	combined KLP18 and LMZ18	similar to above but multi-bit encryption	efficient and decentralization
THL21	MKFHE.Expand (Encode, Link, Decode)	convert single key to multi-key scheme	efficient and decentralization
BD21	symmetric key setting without CRS	similar to above but more efficient	efficient and decentralization

In 2018, Kim et al., based on MW16 [11], constructed a GSW-type KLP18 [10] scheme which discards the CRS in the public parameters. This scheme does not directly extend from single-key to a multi-key. Users use the same single-key FHE algorithm, and the polynomial-time algorithm LinkAlgo will link different users' keys and extend single-key ciphertext into multi-key ciphertext. The single-key encryption step is independent of LinkAlgo, so that when this scheme is used in practice, users can each use single-key encryption and then jointly transform the ciphertext from single-key into multi-key form. They also used the scheme to construct a three-round secure multi-party computation (MPC) protocol against semi-malicious security.

In 2021, Li et al. constructed the LTH21 [21] scheme by combining the advantages of both LMZ18 [22] and KLP18 [10]. They obtained a CRS-free multi-bit encryption MKFHE by using the LinkAlgo algorithm in KLP18 and a plaintext matrix in LMZ18. The former realized the discarding of CRS and a single-key to multi-key transformation, and the latter realized multi-bit encryption.

In 2021, inspired by the KLP18 scheme, Tang et al. also did not extend the single-key, GSW, fully homomorphic scheme to a multi-key one, but designed the algorithm MFHE.Expend (encode, link, decode) based on the original single-key, GSW, FHE scheme to extend the ciphertext of the original scheme from the single-key form to the multi-key form and to link the user's keys, and the THL21 [23] scheme was obtained. In terms of memory and noise, the KLP18 scheme requires each participant to compute and store numbers of n×m dimensional matrices when generating the extended ciphertext, and its final decryption noise is $2(m^4 + m)mNB_\chi$. The THL21 scheme requires only one dimensional matrix and N numbers of dimensional matrices, and its decryption noise is $(2 + m)mNB_\chi$. The THL21 scheme is more efficient and less costly compared to the KLP18 scheme.

In 2021, Biswas et al., constructed the BD21 [24] scheme based on PS16 [25], which reduces the overhead by discarding CRS making the ciphertext extension require only one component (less than PS16) while retaining the advantages of PS16: it is dynamic and multi-hop; there is no necessity to set the number of participants; any user can participate in the operation; and it is bootstrap free. The sizes of BD21's public extension key and extension ciphertext exceed those of PS16, MW16, KLP18, and CCS19, but it does not require a public random matrix, unlike PS16, MW16, and CCS19 [26], and it is multi-hop, unlike KLP18.

2.4. MKFHE for Multi-Bit Encryption

The original multi-key fully homomorphic scheme can only support single-bit encryption. When it comes to dealing with a large amount of data in practice, the encryption step needs to be performed repeatedly. Additionally, the single-bit encrypted ciphertext makes the homomorphic operation inefficient. Therefore, MKFHE schemes that implement multi-bit encryption can effectively improve efficiency. The schemes presented in this section all implement multi-bit encryption, and some of them support batch processing techniques.

The summary of Section 2.4 is shown in Table 3.

Table 3. Summary of Section 2.4.

Scheme	Technology	Function	Emphasis
CZW17	first BGV MKFHE	encrypt ring plaintext	efficient
LMZ18	ciphertext packing	build a plaintext matrix	efficient
LJ20	gadget vector and bit decomposition	remain ciphertext size unchanged	efficient and cost

In TCC2017, Chen et al. proposed the first BGVtype multi-hop MKFHE scheme, CZW17 [12], with security based on ring learning with error (RLWE). The previous GSW-type MKFHE schemes, such as CM15 [27], MW16 [11], BP16 [20], and PS16 [25], although they also have a version based on RLWE, are only capable of encrypting single-bit ciphertexts due to the property of GSW13. The CZW17 scheme is constructed based on the BGV FHE [19]. On the other hand, it can encrypt ring elements rather than single bits. Thereafter, in 2019, Ningbo Li optimized the ciphertext extension of CZW17 to obtain the LZY+19 [28] scheme. Chen et al. optimized the relinearization step of LZY+19 to construct the CDKS19 [29] scheme. In 2021, Yang et al. optimized the relinearization process of CDKS19 to obtain the YZZ22 [30] scheme. These schemes all support multi-bit encryption. In addition, all these schemes are able to realize batch processing using the Chinese residue theorem [31].

In 2018, Li et al. successfully constructed a GSW-type MKFHE scheme LMZ18 [22] based on MW16 that supports multi-bit encryption by using the "ciphertext packing" technique [32] to build a plaintext matrix by embedding the plaintext into the message matrix. Both encryption and ciphertext expansion operations are performed on the basis of the plaintext matrix to achieve the goal of multi-bit encryption. The LTH21 scheme above uses the same method to realize multi-bit encryption.

In 2020, Li et al. proposed NTRU-type MKFHE scheme LJ20 [33] that supports encrypt ring elements. They replace relinearization with gadget vector and bit decomposition techniques [34]. This scheme is capable of batch processing by the Chinese remainder theorem (CRT). No relinearization or ciphertext expansion is required for the ciphertext size to remain unchanged. This scheme is a hierarchical MKFHE, which also needs to be transformed into a full MKFHE using Gentry's bootstrapping theorem [35].

2.5. MKFHE with Cipher Processing Optimization

There is much room for optimization in the processing of ciphertext. Both ciphertext extension and joint decryption have efficiency and overhead issues. Optimizing the extension method by reducing the size of the extended cipher and implementing compressible

ciphertexts are solutions to the problem. This section introduces schemes that optimize the processing of ciphertext.

The summary of Section 2.5 is shown in Table 4.

Table 4. Summary of Section 2.5.

Scheme	Technology	Function	Emphasis
CZW17	zero-padded ciphertext vector	reduce complexity of ciphertext expansion	efficient
LZY+19	nested ciphertext extension	reduce extended ciphertext size	efficient and cost
CCS19	a new ciphertext multiplication	reduce evaluated ciphertext size	efficient and cost
ZZC20	compact ciphertexts	reduce ciphertext size	efficient and cost
SWC21	compressible ciphertexts	compress multiple ciphertexts into one	cost
CDL21	distributed ciphertext extension		efficient and cost
YZZ22	rescaling techniques	reduce the complexity of relinearization	efficient

The CZW17 [12] scheme proposed by Chen et al. in 2017 not only has the advantage of being able to encrypt ring elements, but also improves on ciphertext expansion. The ciphertext expansion is realized by using a zero-padded ciphertext vector, and the computational complexity of the ciphertext expansion is not related to the sizes of ciphertexts, but only to the number of keys involved in the process. Chen also used this scheme to construct a 2-round MPC that supports threshold decryption.

In 2019, Li et al. obtained the LZY+19 [28] scheme by optimizing CZW17. They proposed a nested ciphertext extension method to reduce the size of the evaluation key and the extended ciphertext. They also designed a directed decryption protocol that allows any user to access the decryption results, not limited to those participating in homomorphic evaluation. In the same year, Chen et al. obtained the CDKS19 scheme by optimizing the relinearization procedure of the LZY+19 scheme. LZY+19 does not need ciphertext expansion, which greatly improves efficiency. Chen applies it in privacy protection in neural networks.

In 2019, Chen et al. proposed the MKFHE scheme CCS19 [26] based on the framework of Chillotti et al. (2017) (CCGI17), which can evaluate any binary gate on the encryption bits and then bootstrap. It uses two methods to multiply single-key encryption by multi-key RLWE ciphertexts and controls the growth of the ciphertext size with the ciphertext length, which is only linearly related to the number of participants. The MKFHE software library TFHE, which is an important guide, was also written.

In 2020, Zhou et al. proposed the general MKFHE construction ZZC20 [36] which has compact ciphertexts and specifically two MKFHE schemes (BGV type and TFHE type) with compact ciphertexts. In this construction, a joint secret key, also called a compact key, whose length is independent of the number of parties, is constructed by accumulating the secret keys of the parties. They obtained a joint ciphertext by designing a new ciphertext extension algorithm, whose length is also irrelevant to the number of parties involved in encryption. As a result, the efficiency of homomorphic computation for this general scheme is comparable to that of a single-key FHE scheme. In addition, the user needs to be authorized to add the ciphertext to the homomorphic operation; i.e., all parties involved need to regenerate their cumulative evaluation keys.

In 2019, Gentry et al. proposed a ciphertext compression method that can compress multiple ciphertexts into a single compressed ciphertext, which can reduce communication consumption, and specifically constructed a GSW-type single-key fully homomorphic scheme, GH21 [37], with compressible ciphertexts.

In 2021, Shen et al. proposed the MKFHE scheme SWC21 [14] with compressible ciphertexts based on the single-key, fully homomorphic ciphertext compression scheme proposed by Gentry et al. They applied this method to MKFHE. They removed the unit

matrix in the private key and modified the structure of the extended ciphertext to conform to the conditions of use of the pseudo-square tool array proposed by Gentry when extending the matrix version of the original GSW, fully homomorphic scheme to MKFHE. This scheme can compress the obtained ciphertext and reduce communication cost.

In 2021, Chen et al. constructed a dynamic NTRU-type MKFHE scheme CDL21 [38] based on the LWE assumption in the public key setting of requiring less local random access memory (RAM). The original dynamic MKFHE ciphertext extension and homomorphic operations are performed on the cloud, which requires higher computational power for the cloud and more fees to be delivered to the cloud service provider. Therefore, Chen et al. designed a distributed ciphertext extension method that allows participants to interact over the Internet to perform the ciphertext extension process, and the cloud only undertakes homomorphic computations, reducing the work of the cloud and improving the efficiency of ciphertext extension.

In 2022, Yang et al. obtained the YZZ22 [30] scheme by optimizing the relinearization process of CDKS19. Instead of using gadget vectors to decompose the public key and extended ciphertext, they chose to increase the modulus and use rescaling techniques to realize relinearization. However, this also leads to an increase in errors, which needs to be reduced by decreasing the modulus after key switching.

The comparison of all selected schemes is shown in Table 5.

Table 5. Comparison of selected schemes.

Scheme	Assumption	pk	CT	CRM	Dynamic	Bootstrap	Batch
CM15	LWE	n^2d^2	$k^2n^2d^2$	yes	no	no	no
CM15	RLWE	nd^2	k^2nd^2	yes	no	no	no
BP16	LWE	n^3	nk	yes	yes	yes	no
MW16	LWE	nd^2	$n^2k^2d^2$	yes	no	yes	
PS16 1	LWE	$n(K+d)^2$	$n^3k(K+d)^4$	yes	yes	no	no
PS16 2	LWE/KDM	$n^4(K+d)^4$	$n^2k^2(K+d)^2$	yes	yes	no	no
KLP18	LWE	nd^2	$n^2k^2d^2$	no	no	yes	no
CCS19	RLWE&Circular Security	n^2k^2	nk	yes	no	yes	no
CZL20	RLWE	$nk\,(K+d)$	$n(K+d)^2$	no	no	no	no
BD21	LWE	$n^3(K+d)^2$	$n^2k^2(K+d)^2$	no	yes	no	no
CZW17	LWE	n^3d^7	knd	yes	yes	no	no
CZW17	RLWE	n^2d^6	kd	yes	yes	no	yes
CDL21	LWE	$n^3(K+1)^2$	$n^2k^2(K+d)^2$	yes	yes	no	no
CDKS19	RLWE	nk	nk	yes		yes	yes
LZY+19	GLWE	nk^3	nk	yes	no	yes	yes
ZZC20	GLWE	nk	nk	yes		yes	
HWC21	LWR	k^3	nk	yes	yes		no
YZZ22	RLWE	nk	nk	yes	no	yes	no
CZY21	RLWE			yes		yes	no
LJ20	RLWE&DSPR	n^2Kd^3	n^2Kd^4			yes	yes

3. Optimization Techniques for MKFHE

In this section, two techniques are introduced that the authors believe should be investigated in depth. One is the LinkAlgo algorithm that allows the non-CRS setting of MKFHE, and the other is the compressible ciphertext in SWC21. The former is the first

MKFHE scheme to be constructed in a non-CRS setting, focusing on and strengthening the user's individual key generation ability. Previously, MKFHE could not avoid the distribution of CRS, and the user's personal ability to generate keys was limited. On the one hand, this technique is introduced here in the hope that other researchers can be inspired to develop a more efficient MKFHE without CRS or other ways to enhance the user's ability to generate keys by himself. The ciphertext compression method used by the latter breaks through the compression ratio of 1/2 of the ciphertext size reduction technique used by MKFHE. Although this technique has strict requirements on the ciphertext structure, it is extremely restrictive. However, this idea can be extended for other schemes based on it to involve an efficient compression method combined with its own optimization. Hopefully, these two techniques will be enlightening.

3.1. LinkAlgo Algorithm for the KLP18 Scheme

When the KLP18 scheme extends the single-key GSW-type FHE scheme into MKFHE, firstly, the public random string in the public parameters is dispensed with, and secondly, the single-key scheme is independent of the LinkAlgo algorithm designed by it, so that the user only needs to use the single-key scheme for encryption, and finally, the users involved in the operation will jointly extend it into a multi-key ciphertext through the LinkAlgo algorithm and carry out joint decryption, and its design. The algorithm and the idea of its design provide a new direction for the design of the MKFHE scheme, which has great referential significance. Therefore, the LinkAlgo algorithm is introduced here.

3.1.1. Notion of LinkAlgo

The lowercase bold letters denote vectors, and the uppercase bold letters denote matrices. $\mathbf{x}$ is the column vector, $\mathbf{x}^T$ is the row vector. $\mathbf{A}$ is a matrix, $\mathbf{A}^{(i,j)}$ denote the i-th row and j-th column element of matrix $\mathbf{A}$, $\mathbf{A}_j^{col}$ denotes the j-th column of the matrix, and $\mathbf{A}_i^{row}$ denotes the i-th row of the matrix. $[\mathbf{A} \mid \mathbf{Ax}]$ denotes the horizontal connection of a vector or matrix.

Theorem 1 ([35])**.** *To any $m > n\lceil logq \rceil$, there is a matrix $\boldsymbol{G} \in Z_q^{n\times m}$, its corresponding matrix $\boldsymbol{G}^{-1}(\cdot)$, and matrix $\boldsymbol{M} \in Z_q^{n\times m'}$ (m' is random), which satisfy $\boldsymbol{G}^{-1}(\boldsymbol{M}) \in 0,1^{m\times m^{-1}}$ and $\boldsymbol{G}\boldsymbol{G}^{-1}(\boldsymbol{M}) = \boldsymbol{M}$.*

Theorem 2 ([23])**.** *Set $m \geqslant nlogq + 2\lambda, n \in N, q \in N$. χ is a discrete Gaussian distribution over Z which makes LWE a hard problem. $t = O(logn)$ is an integer. Define two distributions X and Y as follows:*

X is $m \times n$ distributed matrices. $X = \boldsymbol{A} = [\bar{\boldsymbol{A}} \mid \boldsymbol{b}_1 \mid \cdots \mid \boldsymbol{b}_t] = [\bar{\boldsymbol{A}} \mid \boldsymbol{u} - \bar{\boldsymbol{A}}\bar{\boldsymbol{t}}_1 \mid \cdots \mid \boldsymbol{u} - \bar{\boldsymbol{A}}\bar{\boldsymbol{t}}_t]$. $\bar{\boldsymbol{A}} \in Z_q^{n\times\bar{m}}$ is randomly selected. When $1 \leq i \leq t$, $\boldsymbol{b}_i = \boldsymbol{u} - \bar{\boldsymbol{A}}\bar{\boldsymbol{t}}_i mod q \in Z_q^{n\times 1}$. $\bar{\boldsymbol{t}}_i$ is drawn from a Gaussian discrete distribution $\chi^{\bar{m}\times 1}$. Y is a uniform distribution over $Z_q^{n\times 1}$. Then, X and Y are computationally indistinguishable.

Definition 1. *Suppose a distribution $\chi_{n_{n\in N}}$ is based on a distribution of integers. If the distribution satisfies the following property:*

$$Pr_{x\leftarrow\chi_n}[\mid x \mid \geq B] = negl\ (\lambda),$$

then the distribution is called B-bounded.

Definition 2. *β-noisy ciphertext: A ciphertext C that encrypts m' under a private key $\tilde{t}$ is called β-noisy ciphertext. $\tilde{t}^T \cdot C = error + \tilde{t}^T \cdot \boldsymbol{M} \cdot \boldsymbol{G}$, $\parallel error \parallel^{\infty} \leqslant \beta$.*

3.1.2. *LinkAlgo* Algorithm

The matrix $\mathbf{M} \in \{0,\ 1\}^{n\times N}$, $\mathbf{C}^{(s,t)}$ is the β-noise ciphertext of $\mathbf{M}^{(s,\ t)}$ encrypted under $(pk,\ sk) = (\mathbf{F},\ \tilde{\mathbf{t}})$ using the GSW encryption algorithm, i.e., $\mathbf{C}^{(s,t)} = F^T M + M^{[s,t]}G$, where $(s \in [n], t \in [N])$. Let $(pk^{'},\ sk^{'})$=$(\mathbf{F}^{'},\ \tilde{\mathbf{t}}^{'})$ be another pair of keys. Inputs $pk^{'}$ and all $\mathbf{C}^{(s,\ t)} \in Z_q^{(m+1)\times N}$ are given to the *LinkAlgo* algorithm, and it outputs $\mathbf{Y}$.

It holds that $\mathbf{S}^T\mathbf{Y} = \mathbf{S}^T\mathbf{F}'^T\mathbf{M} + \mathbf{e}$ and ($\| \mathbf{e} \|_\infty \leq m^3\beta$ is noise)

Algorithm 1 LinkAlgo algorithms.

Input: $pk^{'}, \{\mathbf{C}^{(s,\ t)}\}_{s\in[n],\ t\in[N]}$;
Output: $\mathbf{Y} \in Z_q^{(m+1)\times N}$;
1: Let $\mathbf{K}_{s,\ t} \in Z_q^{(m+1)\times N}$, $s \in [n]$, $t \in [N]$;
2: Output $\mathbf{Y}$=$\sum_{s=1}^{n}\sum_{t=1}^{N}\mathbf{C}^{(s,\ t)}\mathbf{G}^{-1}(\mathbf{K}_{s,\ t}) \in Z_q^{(m+1)\times N}$

A detailed proof that $\mathbf{S}^T\mathbf{Y} = \mathbf{S}^T\mathbf{F}'^T\mathbf{M} + \mathbf{e}$, $\| \mathbf{e} \|_\infty \leq m^3\beta$ holds is given below:

$$\begin{aligned}\mathbf{S}^T\mathbf{Y} &= \sum_{s,\ t}\mathbf{S}^T\mathbf{C}^{(s,\ t)}\mathbf{G}^{-1}(\mathbf{K}_{(s,\ t)})\\ &= \sum_{s,\ t}(\mathbf{S}^T\mathbf{M}^{(s,\ t)}\mathbf{G} + \mathbf{e}_{s,\ t})\mathbf{G}^{-1}(\mathbf{K}_{s,\ t})\\ &= \sum_{s,\ t}\mathbf{S}^T\mathbf{M}^{(s,\ t)}\mathbf{K}_{s,\ t} + \mathbf{e}^{'}_{s,\ t}\\ &= \mathbf{S}^T\sum_{s,\ t}\mathbf{M}^{(s,\ t)}\mathbf{K}_{s,\ t} + \sum_{s,\ t}\mathbf{e}^{'}_{s,\ t}\end{aligned} \tag{2}$$

where $\mathbf{e}_{s,\ t} = \mathbf{S}^T\mathbf{F}'^T\mathbf{M}, \mathbf{e}^{'}_{s,\ t} = \mathbf{e}_{s,\ t} + \mathbf{G}^{-1}(\mathbf{K}_{s,\ t})$ has a norm $\| \mathbf{e}^{'}_{s,\ t} \| \leq m\beta$.

The correctness of $\sum_{s=1}^{n}\sum_{t=1}^{N}\mathbf{M}^{(s,\ t)}\mathbf{K}^{s,\ t} = \mathbf{F}'^T\mathbf{M}$:

$$\begin{aligned}\sum_{s=1}^{n}\sum_{t=1}^{N}\mathbf{M}^{(s,\ t)}K_{s,\ t} &= \sum_{s=1}^{n}\sum_{t=1}^{N}\begin{bmatrix} 0 & \cdots & \mathbf{M}^{(s,t)}\mathbf{F}'^{\mathrm{T}(1,\ s)} & \cdots & 0\\ \vdots & \cdots & \mathbf{M}^{(s,\ t)}\mathbf{F}'^{\mathrm{T}(2,\ s)} & \cdots & \vdots\\ \vdots & \vdots & \cdots & \cdots & \vdots\\ 0 & \cdots & \mathbf{M}^{(s,\ t)}\mathbf{F}'^{\mathrm{T}(\mathrm{N},\ s)} & \cdots & 0\end{bmatrix}\\ &= \sum_{t=1}^{N}\begin{bmatrix} 0 & \cdots & \sum_{s=1}^{n}\mathbf{M}^{(s,\ t)}\mathbf{F}'^{\mathrm{T}(1,\ s)} & \cdots & 0\\ \vdots & \cdots & \sum_{s=1}^{n}\mathbf{M}^{(s,\ t)}\mathbf{F}'^{\mathrm{T}(2,\ s)} & \cdots & \vdots\\ \vdots & \vdots & \cdots & \cdots & \vdots\\ 0 & \cdots & \sum_{s=1}^{n}\mathbf{M}^{(s,\ t)}\mathbf{F}'^{\mathrm{T}(\mathrm{N},\ s)} & \cdots & 0\end{bmatrix}\end{aligned} \tag{3}$$

$$\begin{aligned}&= \sum_{t=1}^{N}\begin{bmatrix} 0 & \cdots & \mathbf{F}_1^{\mathrm{Trow}}\ \mathbf{M}_t^{\mathrm{col}} & \cdots & 0\\ \vdots & \cdots & \mathbf{F}_2^{\mathrm{Trow}}\ \mathbf{M}_t^{\mathrm{col}} & \cdots & \vdots\\ \vdots & \vdots & \cdots & \cdots & \vdots\\ 0 & \cdots & \mathbf{F}_n^{\mathrm{Trow}}\ \mathbf{M}_t^{\mathrm{col}} & \cdots & 0\end{bmatrix}\\ &= \mathbf{F}^{\mathrm{T}}\mathbf{M}\end{aligned} \tag{4}$$

Therefore, $\mathbf{S}^T\mathbf{Y} = \mathbf{S}^T\mathbf{F}'^T\mathbf{M} + \mathbf{e}$, where $\mathbf{e} = \sum_{s=1}^{n}\sum_{t=1}^{N}\mathbf{e}^{'}_{s,\ t}$ has norm $\| e \|_\infty \leq m^3\beta$.

Input public key $pk_1,\ pk_2,\ \ldots,\ pk_t$ and fresh ciphertext $\mathbf{C}_i$ into LinkAlgo algorithm, which outputs the following extended ciphertext:

1. $\{\mathbf{V}^{(s,\ t)}\}_{s\in[n],\ t\in[N]} \leftarrow \{\mathbf{GSW.ENC}\ (\mathbf{M}^{(s,\ t)},\ pk_j)\}_{s\in[n],\ t\in[N]}$
2. Compute $Y_i^j \leftarrow \{\mathbf{Linkalgo}\ (C^{(s,t)},\ pk_j)\}_{s\in[n],\ t\in[N]} j \in$[t]. The extended ciphertext is:

$$\hat{\mathbf{C}}_i = \begin{bmatrix} \mathbf{C}_i - \mathbf{Y}_i^1 & 0 & \dots & 0 & 0 \\ 0 & \mathbf{C}_i - \mathbf{Y}_i^2 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{Y}_i^i & \dots & \mathbf{C}_i & \dots & \mathbf{Y}_i^i \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 0 & \mathbf{C}_i - \mathbf{Y}_i^t \end{bmatrix} \tag{5}$$

This type of scheme allows users to use single-key homomorphic encryption, and homomorphic evaluation can be performed after the ciphertext is extended using the LinkAlgo algorithm. Users can generate secret keys independently, which has application prospects in some scenarios with related requirements.

3.2. SWC21 Ciphertext Compression Algorithm

Gentry proposed an algorithm for compressing ciphertexts for single-key schemes in GH19 [37] in 2019, and Shen et al. extended the algorithm to the MKFHE scheme SWC21 [14] in 2021, which effectively reduces the communication overhead after compressing ciphertexts, and to some extent drives the development of the MKFHE scheme in terms of efficiency and cost improvements.

Notion of SWC21

Definition 3. *DLWE (decisional learning with errors): Positive integers n and q and an error distribution χ over Z. Let $\boldsymbol{A}_{s,\chi}$ denote the distribution $(\boldsymbol{a}, [<\boldsymbol{a}, \boldsymbol{s}> -2e]_q)$ on $Z_q^n \times Z_q$, where $\boldsymbol{s} \xleftarrow{\$} Z_q^n$, $\boldsymbol{a} \xleftarrow{\$} Z_q^n$ and $e \longleftarrow \chi$. Given m = poly(n) mutually independent instances, these instances are chosen either from the uniform distribution $Z_q^n \times Z_q$ or from the distribution $\boldsymbol{A}_{s,\chi}$.*

Definition 4. *MLWE (matrix learning with errors): Positive integers n, m, r, and q, and an error distribution χ over Z. The matrix learning with errors is to distinguish two distributions. One is $(\boldsymbol{B}, \boldsymbol{A} = \boldsymbol{SB}+\boldsymbol{E})$, where $\boldsymbol{S} \xleftarrow{\$} Z_q^{n\times r}$, $\boldsymbol{B} \xleftarrow{\$} Z_q^{r\times m}$ and $\boldsymbol{E} \longleftarrow \chi^{n\times m}$. Additionally, the other one is uniform distribution $Z_q^{r\times m} \times Z_q^{n\times m}$.*

Definition 5. *Given an integer $q > 2$, for any positive integer $n \in Z^+$, define $\boldsymbol{G}_n \triangleq \boldsymbol{I}_n \otimes \boldsymbol{g}^T \in Z_q^{n\times n\lfloor logq \rfloor}$, where $\boldsymbol{g}^T = [1, 2, 2^2, \cdots, 2^{\lfloor logq \rfloor -1}]$. The symbol $\boldsymbol{G}_n$ is used to denote this matrix in SWC21.*

Lemma 1 ([35])**.** *Positive integers n, m_0, m_1, m, q, and l; $q = q(n)$; $l = \lfloor logq \rfloor$; $m_0 = nl + O(n)$; $m_1 = nl$; and $m = m_0 + m_1$. To $\boldsymbol{A}_0 \xleftarrow{\$} Z_q^{n\times m_0}$, invertible matrix $\boldsymbol{H} \in Z_q^{n\times n}$ and $\boldsymbol{R} \longleftarrow \boldsymbol{D}^{m_0\times m_1}$, there is an efficient randomization algorithm* GemTrap($\boldsymbol{A}_0$,$\boldsymbol{H}$)*, which can generate a matrix $\boldsymbol{A} \triangleq [\boldsymbol{A}_0 \parallel \boldsymbol{HG}_n - \boldsymbol{A}_0\boldsymbol{R}] \in Z_q^{n\times m}$ and a trapdoor $\boldsymbol{R}$, and label $\boldsymbol{H}$. $\boldsymbol{A}$ is not uniformly distributed.*

Lemma 2 ([35])**.** *Given random matrix $\boldsymbol{A} \in Z_q^{n\times n'}$, an efficient randomization algorithm can extract a sub-Gaussian matrix $\boldsymbol{X}$ over $Z_q^{nl\times n'}$ with O(1) as the parameter, which has $\boldsymbol{X} = \boldsymbol{G}_n^{-1}(\boldsymbol{A})$.*

A new technique, the nearly square tool matrix L, is used in GH19 and is also required in SWC21. An open trapdoor matrix $\mathbf{L}^{-1}(0) = \mathbf{F}$ satisfying:
1. **F** has small entries ($\ll q$)
2. $\mathbf{L} \times \mathbf{F}$ = 0(mod q), i.e., all row vectors of **L** can generate a kernel space of **F** mod q;
3. **F** is full-rank over **R**.

When they apply this technology to multi-key version scheme, they adjust the structure of private key and the extended ciphertext. The identity in the private key matrix was deleted so that the extended private key would turn from $s = [i_n s_1 i_n s_2]$ into $s = [i_n s_1 s_2]$, which is a nearly square matrix. Additionally, the extended ciphertext would be split into

$(2+1)^2$ parts rather than 2^2 parts. Meanwhile, there is some information attached to the ciphertext. The extended ciphertext would be like:

$$\hat{C} = \begin{bmatrix} C_{1,\,1} & C_{1,\,2} & D_1 \\ C_{2,\,1} & C_{2,\,2} & D_2 \\ 0 & 0 & C_{3,\,3} \end{bmatrix} \tag{6}$$

- **Initialization Setup** $(1^k,\ 1^N)$: Let k be the security parameter for a large module q with an error distribution $\chi = \chi(k,N)$ bounded by β_χ. Additionally, taken $t \triangleq (t^{'}-1)N$ for the *Comp* step.

  Let $\mathbf{L}^{'} \times \mathbf{F}^{'} = 0$, $\mathbf{L} \triangleq \mathbf{L}^{'} \bigotimes \mathbf{I}_{rN} \in Z_q^{n\times\bar{n}}$ and $\mathbf{F} \triangleq \mathbf{F}^{'} \bigotimes \mathbf{I}_r N \in Z_q^{\bar{n}\times\bar{n}}$ (where $\mathbf{L}^{'} \in Z_q^{(t^{'}-1)t^{'}}$ and $\mathbf{F}^{'} \in Z_q^{t^{'}\times t^{'}}$, $\ell \triangleq \ulcorner logq \urcorner$, $n \triangleq tr$, $\bar{n} \triangleq (t+N)r$, $n \triangleq (t+1)r\ell$ and $\bar{m} \triangleq (t+N)r\ell$), choose $\mathbf{B} \xleftarrow{\$} Z_q^{r\times m}$.
  Output params = $(r, N, q, \ell,\ \beta_\chi,\ \chi, t^{'},\ t, n,\ \bar{n},\ m,\ \bar{m},\ \mathbf{L},\ \mathbf{L}^{'},\ \mathbf{F},\ \mathbf{F}^{'},\ \mathbf{B})$.

- **Keygen** (params): Choose $\mathbf{S} \xleftarrow{\$} Z_q^{r\times m}$ and $\mathbf{E} \leftarrow \chi^{tr\times m}$. Let $\mathbf{A} = \mathbf{SB} + \mathbf{E}$, $\bar{\mathbf{S}} \triangleq [\mathbf{I}_{tr} \mid \mathbf{S}] \in Z_q^{tr\times(t+1)r}$ and $\bar{\mathbf{A}} \triangleq \begin{bmatrix} \mathbf{A} \\ \mathbf{B} \end{bmatrix} \in Z_q^{(t+1)r\times m}$, noticed that $\bar{\mathbf{S}\mathbf{A}} = \mathbf{E}$. Output $PK \triangleq \bar{\mathbf{A}}$, $SK \triangleq \bar{\mathbf{S}}$.

- **Enc** $(PK_i,\ \mu)$: Input a plaintext bit $\mu \in \{0,\ 1\}$, choose $\bar{\mathbf{J}}_i \xleftarrow{\$} \{0,\ 1\}^{m\times m}$, let ciphertext be $\bar{\mathbf{C}}_i \triangleq \mu\mathbf{G}_{(t+1)r} + \bar{\mathbf{A}}_i\bar{\mathbf{J}}_i \in Z_q^{(t+1)r\times m}$. $\bar{\mathbf{C}}_i$ will be divided as follows for the *Exp* step:

$$\begin{aligned} \bar{\mathbf{C}}_i^{(1,\,1)} &= \bar{\mathbf{C}}_i[\text{pre tr rows, pre tr}\ell \text{ columns}] \in \mathrm{Z_q^{tr\times tr\ell}}; \\ \bar{\mathbf{C}}_i^{(2,\,1)} &= \bar{\mathbf{C}}_i[\text{last r rows, pre tr}\ell \text{ columns}] \in \mathrm{Z_q^{r\times tr\ell}}; \\ \bar{\mathbf{C}}_i^{(1,\,2)} &= \bar{\mathbf{C}}_i[\text{last tr rows, last r}\ell \text{ columns}] \in \mathrm{Z_q^{tr\times r\ell}}; \\ \bar{\mathbf{C}}_i^{(2,\,2)} &= \bar{\mathbf{C}}_i[\text{last r rows, last tr}\ell \text{ columns}] \in \mathrm{Z_q^{r\times r\ell}}; \end{aligned} \tag{7}$$

  Let $\mathbf{J}_i = \bar{\mathbf{J}}_i[\text{the last r}\ell \text{ columns}] \in \{0,1\}^{\mathrm{m\times r\ell}}$.

  Notice that $\bar{\mathbf{C}}_i^{(2,\,2)} = \mu\mathbf{G}_r + \mathbf{B}\mathbf{J}_i$.

  Attached information $\mathbf{O}_i \triangleq (\mathbf{V}_{i,\,1}^{(x,\,y,\,b)}, \mathbf{V}_{i,\,2}^{(x,\,y,\,b)})_{x\in[t],\ y\in[m],\ b\in[rl]}$ is needed to successfully execute the Exp step.

$$\mathbf{V}_{i,\,1}^{(x,\,y,\,b)} = -\mathbf{A}_i\tilde{\mathbf{J}}_i^{(x,\,y,\,b)} + \mathbf{J}_i[y,\ b]\mathbf{G}_m^{(x)} \in Z_q^{tr\times r\ell}; \tag{8}$$

$$\mathbf{V}_{i,\,2}^{(x,\,y,\,b)} = \mathbf{B}\tilde{\mathbf{J}}_i^{(x,\,y,\,b)} \in Z_q^{r\times r\ell}; \tag{9}$$

  where $\bar{\mathbf{J}}_i^{(x,\,y,\,b)} \triangleq \{0,\ 1\}^{m\times r\ell}$, $\mathbf{G}_r(x) = \begin{bmatrix} 0^{(x-1)r\times r\ell} \\ \mathbf{G}_r \\ 0^{(t-x)r\times r\ell} \end{bmatrix}$. The complete ciphertext tuple is $C_i \triangleq (\bar{C},\ O_i)$

- **Exp** $(id_i \in [N],\ C_i)$: extended ciphertext is:

$$\bar{\mathbf{C}}_i \triangleq \begin{bmatrix} \bar{\mathbf{C}}_i^{(1,\,1)} & \mathbf{D}_{i,\,1}^{(1)} & \dots & \bar{\mathbf{C}}_i^{(1,\,2)} & \dots & \mathbf{D}_{i,\,N}^{(1)} \\ & \bar{\mathbf{C}}_i^{(2,\,2)} & & & & \\ & & \ddots & & & \\ \bar{\mathbf{C}}_i(2,\ 1) & \mathbf{D}_{i,\,1}^{(2)} & \dots & \bar{\mathbf{C}}_i^{(2,\,2)} & \dots & \\ & & & & \ddots & \\ & & & & & \bar{\mathbf{C}}_i^{(2,\,2)} \end{bmatrix} \in Z_q^{\bar{n}\times\bar{m}} \tag{10}$$

Fresh ciphertext does not support compression. Therefore, the ciphertext of different users needs to be pre-processed before compression, which is similar to the traditional GSW-expand step.

(1) Generate components required for extended ciphertext:

$$\mathbf{Z}_{i,j}^{(x,\,y,\,b)} \triangleq \sum_{z_1=1}^{r} -\mathbf{A}_j[z_1 + (x-1)r,\ y] \cdot \mathbf{E}'_{z_1,\,b} \in Z^{r\times r\ell} \tag{11}$$

where $\mathbf{E}'_{z_1,\,b} \in Z^{r\times r\ell}$ and only the intersection of column b and row z_1 is 1; all others are 0.

Let

$$\bar{\mathbf{Z}}_{i,\,j}^{(x,\,y,\,b)} \triangleq \begin{bmatrix} 0^{(x-1)r\times r\ell} \\ \mathbf{Z}_{i,\,j}^{(x,\,y,\,b)} \\ 0^{(t-x)r\times r\ell} \end{bmatrix} \in Z_q^{tr\times r\ell} \tag{12}$$

(2) Generate auxiliary ciphertext $X_{i,j}^s, s \in (0,1)$:

$$\mathbf{D}_{i,\,j}^{(1)} \triangleq \sum_{x=1}^{t}\sum_{y=1}^{m}\sum_{b=1}^{rl} \mathbf{V}_{i,\,1}^{(x,\,y,\,b)} \cdot \mathbf{G}_r^{-1}(Z_{i,\,j}^{(x,\,y,\,b)}) \in Z_q^{tr\times r\ell}; \tag{13}$$

$$\mathbf{D}_{i,\,j}^{(2)} \triangleq \sum_{x=1}^{t}\sum_{y=1}^{m}\sum_{b=1}^{rl} \mathbf{V}_{i,\,2}^{(x,\,y,\,b)} \cdot \mathbf{G}_r^{-1}(Z_{i,\,j}^{(x,\,y,\,b)}) \in Z_q^{r\times r\ell}; \tag{14}$$

- **Comp** (params,$\{\hat{\mathbf{C}}_{u,\,v,\,w}\}_{u,\,v\in[n],\,w\in[\ell]}$): Compress one or some ciphertext into ciphertext of a smaller size. Let $\mathbf{T}_{O,\,V} \triangleq \begin{bmatrix} \mathbf{E}'_{u,\,v} \\ 0^{Nt\times n} \end{bmatrix} \in Z^{\bar{n}\times n}$, where $\mathbf{E}'_{u,\,v} \in Z^{n\times n}$ and only the intersection of the u-th row and v-th column are 1; all others are 0. Let compressed ciphertext be:

$$\mathbf{C}^* \triangleq \sum_{u\in[n]}\sum_{v\in[n]}\sum_{w\in[\ell]} \hat{\mathbf{C}}_{u,\,v,\,w} \times \mathbf{G}_{\bar{n}}^{-1}(2_w \cdot \mathbf{T}_{u,\,v} \times \mathbf{L}) \in Z_q^{\bar{n}\times\bar{n}}. \tag{15}$$

The compressing algorithm is the same as Gentry's GH19 scheme; all the cipher processing is to meet the requiements of this algorithm.

- **Eval** (params, f, $\hat{\mathbf{C}}_1, \hat{\mathbf{C}}_2, \dots, \hat{\mathbf{C}}_s$): Input N and circuit $f : \{0,1\}^t \to \{0,1\}$ and a string of ciphertext $\hat{\mathbf{C}}_1, \hat{\mathbf{C}}_2, \dots, \hat{\mathbf{C}}_s$; output the ciphertext $\hat{C}_f$ after homomorphism.
Eval.add $\triangleq$ $\mathbf{C}_1 + \mathbf{C}_2$;
Eval.mult $\triangleq$ $\mathbf{C}_1 \times \mathbf{G}_{\bar{n}^{-1}} \times \mathbf{C}_2$

- **CompDec** ($\mathbf{C}^*$, $(SK_i)_{i\in[N]}$): Input the private key of N participants; let $\mathbf{S} \triangleq [\mathbf{I}_n, \mathbf{S}_1, \ldots, \mathbf{S}_N] \in Z_q^{\bar{n}\times\bar{n}}$ be the extended private key. Decrypting a compressed cipher involves the following four steps:
  1. $\mathbf{Z} \triangleq \mathbf{S} \times \mathbf{C}^*(modq)$;
  2. $\mathbf{W} \triangleq \mathbf{Z} \times \mathbf{F}(modq)$;
  3. $\mathbf{P} \triangleq \mathbf{W} \times \mathbf{F}^{-1}$, *where* $\mathbf{F} \in Z_q^{\bar{n}\times\bar{n}}$ is the open trapdoor matrix;
  4. $\mathbf{M}' \triangleq (\mathbf{Z} - \mathbf{P}) \times \mathbf{L}^{-1}(modq)$ ($\mathbf{L}$ is full-rank and $\mathbf{L} \times \mathbf{L}^{-1} \in Z_q^{\bar{n}\times\bar{n}} = \mathbf{I}_n$).

4. Application of MKFHE

As the current cloud environment is booming, there are increasingly many scenarios where multiple users are involved in the computation. Users want to participate in the computation to get the result, but at the same time they do not want to share their data with others. MKFHE has the characteristics of homomorphic computation between ciphertexts under different keys and joint decryption by participating users, which is very suitable for the requirement of privacy protection [39,40]. MKFHE has been applied in the following applications:

4.1. Medical Applications

For medical research, comprehensive and rich patient data can facilitate the progress of medical practice. Adequate patient data can be used to build predictive models for rapid detection of disease causes and timely treatment, e.g., for disease-causing gene localization [41], which requires access to a sufficient amount of patient genetic information from different medical institutions.

Model of MKFHE's Application to Medical Data

Patient privacy issues cannot be avoided in medical research. Therefore, in this model, the medical-information-sharing organization composed of medical institutions carries out secure personal data sharing according to the following model when the researcher initiates the request to query data.

In this model, we add a homomorphic accelerator next to the aggregation server. Some researchers take advantage of the parallel computing capabilities of high-performance computing platform architectures (GPUs and FPGA) to deal with the numerous repeated and complex operations in homomorphic evaluation to improve data throughput and computing parallelism. The methods used include, but are not limited to, using mathematical theorems to convert operations into a form suitable for parallel computing and designing specialized homomorphic evaluation hardware (FPGA). The aggregation server can hand over the complex homomorphic evaluation process to the homomorphic accelerator to improve efficiency.

1. After each medical institution receives the request, locally compute the result of query; then encrypt the result and broadcast.
2. Each medical institution aggregates encrypted results locally. For iterative tasks, each medical institution repeats the process until the requirements are met.
3. The public key of the final result switched from collective public key to the public key of querier through collective key switching.
4. The querier decrypts the final result.

The model of MKFHE on medical is shown in Figure 3.

4.2. Financial Scenarios

Any service in the financial industry is based on credit. Financial institutions need to evaluate all aspects of the client to complete the user's image. This involves several different data sources, and the data providers are responsible for the client's privacy and comply with the relevant privacy protection laws while sharing clients' data. MKFHE can

be applied to create a privacy-protected customer credit evaluation system [42] and perfect the current credit system.

Model of MKFHE's Application to Financial Data

In this model, banks form a data sharing organization. Each bank registers with the TA (trusted authentic) and negotiates the signature key. Each bank obtains a pseudonymous ID, and the real ID is stored in the TEE (trusted execution environment) by the TA. When a bank needs to use the data of other banks to evaluate the user's credit, it initiates a data sharing request. Additionally, due to the time-fake ID used at request, none of the parties know the real identity of the initiator. After completing the data sharing, this data sharing will be recorded in the blockchain. The specific data sharing process is as follows:

After completing the registration and KeyGen of MKFHE, each bank holds the PID (pseudonym ID), signing key, and MKFHE key pair. The bank initiates the data sharing request and sends the request, PID, and digital signature to the TA. The TA locally retrieves the users that match the request and forms them into a data sharing group.

1. Each bank in the group trains the model on the local data and encrypts the obtained gradient using the public key of MKFHE. After signing the encrypted gradient, it is sent to other users.
2. After receiving the information, each user verifies the validity of the information via digital signature. If valid, then the local model is obtained by training together with the local gradient.
3. The local model is sent to the AS (aggregation server), and the corresponding block is generated in the blockchain for record keeping. The AS aggregates all the models and broadcasts them to the users within the group.
4. Each user updates the local model with the global model. We repeat steps three and four until the accuracy condition is met. Then, each user performs partial decryption of MKFHE to obtain the plaintext result.

When a malicious behavior occurs, the trusted third party can retrieve the real identity of the malicious user in the local trusted execution environment (TEE). The malicious user's identity is then broadcast to all participants and included in the list of malicious users.

The model of MKFHE on financial is shown in Figure 4.

4.3. IoT

The IoT is usually combined with cloud computing, which connects physical terminals through the cloud and gives life to data. Physical terminals include personal devices, such as mobile phones, tablets, and cars, and important devices related to national security, such as national power grids and military drones, which are inevitably threatened by hackers during data collection, transmission, and use. However, most of these schemes use single-key homomorphic encryption [43–45], and there is only one recent study that used multi-key homomorphic encryption in a scenario where federal learning was combined with IoT [46].

Model of MKFHE's Application on IoT

Multi-key fully homomorphic encryption can be applied to most multi-user scenarios for privacy protection. Take Iot as an example. Iot can be combined with federated learning to protect the privacy of edge devices. In this structure, local data of edge devices do not come out; only the model parameter which was trained from local data will be shared. However, the attackers may infer from the model parameters of participants' information. Therefore, MKFHE can be used to encrypt local model parameters to further enhance the data security of edge devices.

Preparation: Initialize MKFHE, and each party has obtained their own public and secret key pair.

1. Each device uses the downloaded global model to train the local data to get the local model and uses its own public key pk_i to encrypt the local model parameter; then, it sends the encrypted parameter to the aggregation server.
2. After receiving the encrypted model parameters of all parties, the aggregation server uses the aggregation algorithm to get the average model. The aggregation server sends the average model parameter to all parties.
3. After receiving the average model parameter, all parties use it to update the local model.

Repeat the above steps until the preset stop condition is met.

After meeting the stop condition, each participant can use their own private key for partial decryption, and the final decryption result can be obtained after all partial decryption is summarized.

The model of MKFHE on Iot with federated learning is shown in Figure 5.

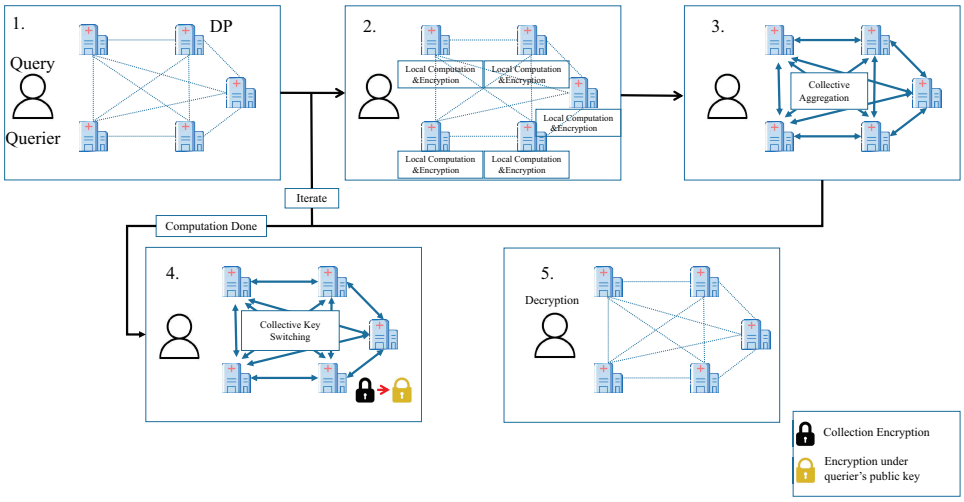

Figure 3. Application of MKFHE on medical.

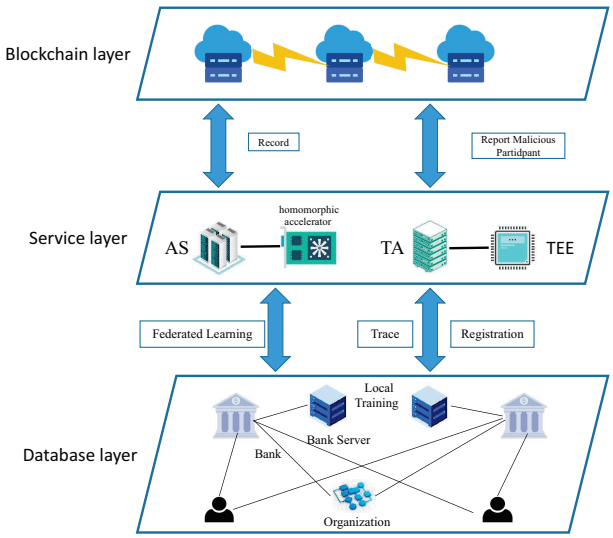

Figure 4. Application of MKFHE on financial data.

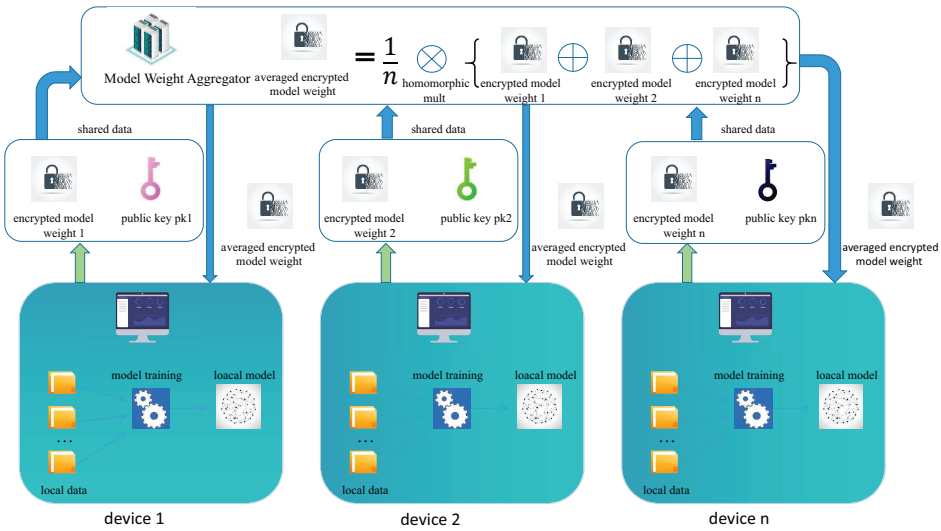

Figure 5. Typical application of MKFHE on Iot with federated learning.

4.4. Service Recommendation

Product recommendations, user matching, and friend recommendations in web services need to use data such as users' personal information and browsing history, which makes many users feel their privacy is being violated. MKFHE can perform operations with data encrypted into ciphertext. Service providers can combine it with their technology to provide services to users without disclosing their data, such as online car matching [47], Drip, Uber, and other taxi-hailing software, which require access to the user's location information, which means the service provider can obtain or infer the user's address or work address. With the help of MKFHE, a taxi matching system can effectively hide the user's personal information while completing the match.

4.5. Research Innovation

MKFHE technology has been introduced for a short period of time, and there is still great room for improvement. Its multi-key feature meets the requirements of the cloud environment, so there are considerable prospects for innovation—for example, combining MKFHE with technologies such as federation learning [48,49], secure multi-party computing [50], deep learning [51,52], and blockchain [53,54] to enhance the properties of privacy protection and build solutions that meet the current emphasis on personal information protection policy requirements.

The potential of MKFHE does not stop with these cases. As long as there are multi-user scenarios, MKFHE has the potential to be applied.

5. Conclusions

The urgent need for privacy protection and the birth and development of quantum computers has stimulated the need for protection methods against quantum attacks. This demand promotes the development of MKFHE, which provides theoretical support for future secure data sharing among multiple users. Currently, the complex operation of MKFHE makes it inefficient and costly to apply. Therefore, MKFHE only accounts for a small part of the application scheme of homomorphic encryption on the Internet of Things. As we have discussed at the end of Section 4.1, the application of hardware for acceleration has the potential to enable the application of multi-key fully homomorphic encryption.

In addition to accelerated homomorphic encryption, solutions such as KLP18 and THL21, where each user uses a single key encryption and the process of converting the public extended cipher into a multi-key cipher that is then placed in the cloud, may also be a solution. As the size of the extended ciphertext becomes larger, and the cloud is undertaking the process of ciphertext extension, which can reduce the loss of bandwidth.

Therefore, in the authors' opinion, users only need to upload encrypted data, and the cloud with the corresponding homomorphic hardware accelerator will be set to undertake a large number of homomorphic operations. If possible, acceleration hardware based on a trusted execution environment can be used to further enhance security.

Author Contributions: Conceptualization M.Y., D.W., F.Z., S.W., S.J. and Y.R.; methodology M.Y., D.W. and F.Z.; visualization and writing—original draft M.Y.; project administration M.Y., D.W. and Y.R.; supervision S.J.; writing—review and editing M.Y., S.J. and Y.R. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Key R&D Program of China (no. 2021YFB2700500), the National Natural Science Foundation of China (no. 62072249), the National Key R&D Program of Guangdong Province (no. 2020B0101090002), and the Natural Science Foundation of Jiangsu Province (no. BK20200418, BE2020106).

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ren, Y.; Leng Y.; Qi, J.; Sharma, P.K.; Tolba, A. Multiple cloud storage mechanism based on blockchain in smart homes. *Future Gener. Comput. Syst.* **2021**, *115*, 304–313. [CrossRef]
2. Rivest, R.L.; Adleman, L.M.; Dertouzos, M.L. On data banks and privacy homomorphisms. *Found. Secur. Comput.* **1978**, *4*, 169–179.
3. Gentry, C. Fully homomorphic encryption using ideal lattices. In Proceedings of the forty-first annual ACM symposium on Theory of Computing, Bethesda, MD, USA, 1–2 June 2009; pp. 169–178.
4. Ren, Y.; Yan, L.; Cheng, Y.; Jin, W. Secure data storage based on blockchain and coding in edge computing. *Math. Biosci. Eng. MBE* **2019**, *16*, 1874–1892. [CrossRef] [PubMed]
5. Tang, D.H.; Zhu, S.X.; Wang, L.; Yang, H.M.; Fan, J. Fully homomorphic encryption scheme from rlwe. *J. Commun.* **2014**, *35*, 173–182.
6. Brakerski, Z.; Vaikuntanathan, V. Efficient fully homomorphic encryption from (standard) lwe. *SIAM J. Comput.* **2014**, *43*, 831–871. [CrossRef]
7. Ren, Y.; Huang, D.; Wang, W.H.; Yu, X.F. BSMD: A blockchain-based secure storage mechanism for big spatio-temporal data. *Future Gener. Comput. Syst.* **2023**, *138*, 328–338. [CrossRef]
8. Lopez-Alt, A.; Tromer, E.; Vaikuntanathan,V. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In Proceedings of Forty-Fourth the Annual ACM Symposium on Theory of Computing 2012, New York, NY, USA, 19–22 May 2012.
9. Che, X.; Zhou, T.; Li, N.; Zhou, H.; Chen, Z. Modified multi-key fully homomorphic encryption based on ntru cryptosystem without key-switching. *Tsinghua Sci. Technol.* **2020**, *25*, 14–28. [CrossRef]
10. Kim, E.; Lee, H.S.; Park,J. Towards round-optimal secure multiparty computations: Multikey fhe without a crs. In Proceedings of the Australasian Conference on Information Security and Privacy, Wollongong, NSW, Australia, 11–13 July 2018; Springer: Cham, Switzerland, 2018; pp. 101–113.
11. Mukherjee,P.; Wichs D.; Two round multiparty computation via multi-key fhe. In Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, 8–12 May 2016; Springer: Berlin/Heidelberg, Germany, 2016; pp. 735–763.
12. Long, C.; Zhang, Z.; Wang, X. Batched multi-hop multi-key fhe from ring-lwe with compact ciphertext extension. In Proceedings of the Theory of Cryptography Conference, Baltimore, MD, USA, 12–15 November 2017; Springer: Cham, Switzerland, 2017; Volume 10678, pp. 597–327.
13. Brakerski, Z.; Gentry, C.; Vaikuntanathan, V. (Leveled) fully homomorphic encryption without bootstrapping. *ACM Trans. Comput. Theory (TOCT)* **2014**, *6*, 309–325. [CrossRef]
14. Shen, T.; Wang, F.; Chen, K.; Shen, Z.; Zhang, R. Compressible multikey and multi-identity fully homomorphic encryption. *Secur. Commun. Netw.* **2021**, *2021*, 6619476. [CrossRef]
15. Albrecht, M.; Bai, S.; Ducas, L. A subfield lattice attack on overstretched ntru assumptions. In Proceedings of the Annual International Cryptology Conference, Santa Barbara, CA, USA, 14–18 August 2016; Springer: Berlin/Heidelberg, Germany, 2016; pp. 153–178.
16. Zhou, H.; Li, N.; Che, X.; Yang, X. Multi-key fully homomorphic encryption scheme over prime cyclotomic rings. *IET Inf. Secur.* **2021**, *15*, 472–486. [CrossRef]

17. Yu, Y.; Xu, G.; Wang, X. Provably Secure NTRU Instances over Prime Cyclotomic Rings. In Proceedings of the IACR International Workshop on Public Key Cryptography, Amsterdam, The Netherlands, 28–31 March 2017; Springer: Berlin/Heidelberg, Germany, 2017; Volume 10174, pp. 409–434
18. Huang, Y.; Wu, K.; Chen, M. Fully dynamic multi-key fhe without gaussian noise. *IEEE Access* **2021**, *9*, 50639–50645. [CrossRef]
19. Liu, F.H.; Wang, Z. Rounding in the rings. In Proceedings of the Annual International Cryptology Conference, Santa Barbara, CA, USA, 17–21 August 2020; Springer: Cham, Switzerland, 2020; Volume 12171, pp. 296–326.
20. Brakerski, Z.; Perlman, R. Lattice-based fully dynamic multi-key fhe with short ciphertexts. In Proceedings of the Annual International Cryptology Conference, Santa Barbara, CA, USA, 14–18 August 2016; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2016; Volume 9814, pp. 190–213.
21. Li, X.; Tang, C.; Hu, Y. Multi key fully homomorphic encryption system that supports multi-bit encryption. *J. Cryptol. Res.* **2022**, *9*, 248–256.
22. Li, Z.; Ma, C.; Zhou, H. Multi-key fhe for multi-bit messages. *Sci. China Inf. Sci.* **2018**, *61*, 266–277. [CrossRef]
23. Tang, C.; Hu, Y.; Li, X. Three round secure multiparty computation based on multi-key full-homomorphic encryption without crs. *J. Cryptologic Res.* **2021**, *2*, 273–281.
24. Biswas, C.; Dutta, R. Dynamic multi-key fhe in symmetric key setting from lwe without using common reference matrix. *J. Ambient Intell. Humaniz. Comput.* **2022**, *13*, 1241–1254. [CrossRef]
25. Peikert, C.; Shiehian, S. Multi-key fhe from lwe, revisited. In Proceedings of the Theory of Cryptography Conference, Tel Aviv, Israel, 10–13 January 2016; Springer: Berlin/Heidelberg, Germany, 2016; Volume 9986, pp. 217–238.
26. Chen, H.; Chillotti, I.; Song, Y. Multi-key homomorphic encryption from tfhe. In Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, 8–12 December 2019; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2019; Volume 11922, pp. 446–472.
27. Clear, M.; Mcgoldrick, C. Multi-identity and multi-key leveled fhe from learning with errors. In Proceedings of the Annual Cryptology Conference, Santa Barbara, CA, USA, 16–20 August 2015; Springer: Berlin/Heidelberg, Germany, 2015; Volume 9216, pp. 630–656.
28. Li, N.; Zhou, T.P.; Yang, X.Y.; Han, Y.L.; Tu, G.S. Efficient multi-key fhe with short extended ciphertexts and directed decryption protocol. *IEEE Access* **2019**, *7*, 56724–56732. [CrossRef]
29. Chen, H.; Dai, W.; Kim, M.; Song, Y. Efficient multi-key homomorphic encryption with packed ciphertexts with application to oblivious neural network inference. In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, London, UK, 11–15 November 2019; pp. 395–412.
30. Yang, X.; Zheng, S.; Zhou, T.; Liu, Y.; Che, X. Optimized relinearization algorithm of the multikey homomorphic encryption scheme. *Tsinghua Sci. Technol.* **2022**, *27*, 642–652. [CrossRef]
31. Wang, X.; Xu, G.; Wang, M.; Meng, X. *Mathematical Foundations of Public Key Cryptography*; CRC Press: Boca Raton, FL, USA, 2015.
32. Smart, N.P.; Vercauteren, F. Fully homomorphic simd operations. *Des. Cryptogr.* **2014**, *71*, 57–81. [CrossRef]
33. Li, R.Q; Jia, C.F. A multi-key homomorphic encryption scheme based on ntru. *J. Cryptologic Res.* **2020**, *7*, 683–697.
34. Micciancio, D.; Peikert, C. Trapdoors for lattices: Simpler, tighter, faster, smaller. In Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, 15–19 April 2012; Springer: Berlin/Heidelberg, Germany, 2012; Volume 7237, pp. 700–718.
35. Gentry, C.; Sahai, A.; Waters, B. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Proceedings of the Annual Cryptology Conference, Santa Barbara, CA, USA, 18–22 August 2013; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2013; Volume 8042, pp. 75–92.
36. Zhou, T.; Zhang, Z.; Chen, L.; Che, X.; Liu, W.; Yang, X. Multi-key fully homomorphic encryption scheme with compact ciphertext. *Cryptology ePrint Archive* 2021. Available online: https://eprint.iacr.org/2021/1131 (accessed on 6 September 2021).
37. Gentry, C.; Halevi, S. Compressible fhe with applications to pir. In Proceedings of the Theory of Cryptography Conference, Nuremberg, Germany, 1–5 December 2019; Springer: Cham, Switzerland, 2019; Volume 11892, pp. 438–464.
38. Chen, Y.; Dong, S.; Li, T.; Wang, Y.; Zhou, H. Dynamic multi-key fhe in asymmetric key setting from lwe. *IEEE Trans. Inf. Forensics Secur.* **2021**, *16*, 5239–5249. [CrossRef]
39. Ren, Y.; Zhu, F.; Sharma, P.K.; Wang, T.; Wang, J.; Alfarraj, O.; Tolba, A. Data query mechanism based on hash computing power of blockchain in internet of things. *Sensors* **2020**, *20*, 207. [CrossRef] [PubMed]
40. Wu, Q.; Han, Z.; Mohiuddin, G.; Ren, Y. Distributed timestamp mechanism based on verifiable delay functions. *Comput. Syst. Sci. Eng.* **2023**, *44*, 1633–1646. [CrossRef]
41. Zhou, T.; Liu, W.; Li, N.; Yang, X.; Han, Y.; Zheng, S. Secure scheme for locating disease-causing genes based on multi-key homomorphic encryption. *Tslnghua Sci. Technol.* **2022**, *27*, 333–343. [CrossRef]
42. Liu, J.; He, X; Sun, R.; Du, X.; Guizani, M. Privacy-preserving data sharing scheme with fl via mpc in financial permissioned blockchain. In Proceedings of the ICC 2021-IEEE International Conference on Communications, Montreal, QC, Canada, 14–23 June 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 1–6. [CrossRef]
43. Matsumoto, M. and Oguchi, M. Speeding up sensor data encryption with a common key cryptosystem combined with fully homomorphic encryption on smartphones. In Proceedings of the 2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4), London, UK, 27–28 July 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 500–505. Available online: https://ieeexplore.ieee.org/document/9210393/ (accessed on 1 October 2020).

44. Kolsch, J.; Ratzke, A.; Grimm, C.; Heinz, C.; Nandagopal, G. Simulation based validation of a smart energy use case with homomorphic encryption. In Proceedings of the 2019 15th International Conference on Distributed Computing in Sensor Systems, Santorini Island, Greece, 29–31 May 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 255–262. [CrossRef]
45. Qiu, F.; Yu, J.; Zheng, F.; Liang, L; Li, Y. Electric iot perception layer data privacy-preserving using multi identity-based fully homomorphic encryption. In Proceedings of the 2020 IEEE 3rd International Conference on Automation, Electronics and Electrical Engineering, Shenyang, China, 20–22 November 2020; IEEE: Piscataway, NJ, USA, 2020; pp.30–34. [CrossRef]
46. Ma, J.; Naas, S.A.; Sigg, S.; Lyu, X. Privacy-preserving federated learning based on multi-key homomorphic encryption. *Int. J. Intell. Syst.* **2022**, *37*, 5880–5901. [CrossRef]
47. Xiang, K. Location-Preserving Matching Protocol for Ride-Hailing Service Based on Multi-Key Fully Homomorphic Encryption. Master's Thesis, Harbin Institute of Technology, Harbin, China, 2020.
48. Guo, H. Research and Implementation of Federated Learning That Supports Aggregation under Multiple Keys. Master's Thesis, Harbin Institute of Technology, Harbin, China, 2020.
49. Liu, Y. Research on Efficient Communication and Multi-Key Homomorphic Encryption Technology in Hierarchical Federated Learning Environment. Master's Thesis, Beijing Jiaotong University, Beijing, China, 2021.
50. Wang, H.; Feng, Y.; Zhao, L.; Tang, S. A secure multi-party computation protocol on the basis of multi key homomorphism. *J. South China Univ. Technol. Sci. Ed.* **2017**, *45*, 69–76.
51. Ping, L.; Li, j.; Huang, Z.; Li, T.; Gao, C.Z.; Yiu, S.M.; Chen, K. Multi-key privacy-preserving deep learning in cloud computing. *Future Gener. Comput. Syst.* **2017**, *74*, 76–85.
52. Kwabena, O.; Qin, Z.; Zhuang, T.; Qin, Z. Mscryptonet: Multi-scheme privacy-preserving deep learning in cloud computing. *IEEE Access* **2019**, *7*, 29344–29354. [CrossRef]
53. Ren, Y.; Zhu, F.; Wang, J.; Pradip Kumar Sharma, and Uttam Ghosh. Novel vote scheme for decision-making feedback based on blockchain in internet of vehicles. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 1639–1648. [CrossRef]
54. Huang, L.; Xu, L.; Zhu, L.; Gai, K. A blockchain-assisted privacy-preserving cloud computing method with multiple keys. In Proceedings of the 2021 IEEE 6th International Conference on Smart Cloud, Newark, NJ, USA, 6–8 November 2021; IEEE: Piscataway, NJ, USA, pp. 19–25. [CrossRef]