cryptography

MDPI

Review

A Brief Survey on Hardware Implementation of Fully Homomorphic Encryption

Yang Su [1], Kaixuan Zhou [1], Weidong Zhong [1,*], Jianfei Wang [2], Jia Hou [2] and Chen Yang [2]

[1] School of Cryptography Engineering, Engineering University of People's Armed Police, Xi'an 710086, China; wj_suyang@126.com (Y.S.); zkx_2026@126.com (K.Z.)
[2] School of Microelectronics, Xi'an Jiaotong University, Xi'an 710049, China; jfwang@stu.xjtu.edu.cn (J.W.); hjwst0314@stu.xjtu.edu.cn (J.H.); chyang00@xjtu.edu.cn (C.Y.)
* Correspondence: wdeast@163.com

Abstract

Leveraging the favorable properties of cryptographic computation, FHE effectively ensures data availability without visibility, thereby holding broad application prospects in cloud computing security and data privacy protection. However, computational efficiency remains a critical bottleneck that constrains its practical deployment and further development. Consequently, research on hardware implementations of FHE has become a major direction in the cryptographic community. This paper first systematically reviews the research progress of FHE schemes, summarizing and analyzing the characteristics of representative FHE schemes. Subsequently, we survey and analyze hardware research progress and optimization techniques from the perspectives of overall accelerator architecture design, polynomial multiplier design, and integer modular multiplier design, highlighting the main advantages, disadvantages, and common features of different hardware structures. Finally, based on an analysis of existing hardware implementation architectures for FHE, this paper presents the potential deficiencies, summarizes and outlines future research directions and development prospects, aiming to further improve the operational performance of FHE hardware implementations.

Keywords: fully homomorphic encryption (FHE); hardware implementation; accelerator; polynomial multiplier; modular multiplier

1. Introduction

The advent of FHE [1] has provided a theoretical solution for privacy preservation in cloud computing. FHE enables direct computations on ciphertexts, where the decrypted result is equivalent to that obtained by performing the same operations on plaintexts. Endowed with robust ciphertext computation capabilities, FHE can fulfill computational tasks over encrypted data without revealing any sensitive information. The cloud computing privacy protection model based on FHE is depicted in Figure 1. Users encrypt plaintext data and upload it to the cloud in ciphertext form, ensuring data security during transmission and storage. Upon completing ciphertext computations, the cloud returns the encrypted result to the user for decryption, and the decrypted output is identical to the result of direct plaintext computation. In this process, no other parties, including cloud service providers, can decrypt the ciphertexts without accessing the user's private key, thereby FHE fundamentally guarantees the security of cloud computing data throughout transmission, processing, and storage. Notably, most mainstream FHE schemes are constructed

Check for updates

Academic Editors: Ki-Hyun Jung and Luis Javier García Villalba

Received: 3 June 2026
Revised: 4 July 2026
Accepted: 8 July 2026
Published: 13 July 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

based on lattice-based hard problems. Compared with traditional cryptosystems, FHE possesses quantum attack resistance. As cloud computing privacy protection technologies advance, fully homomorphic cryptosystems have become increasingly pivotal in modern data privacy preservation, and have been widely applied in fields such as ciphertext information retrieval [2,3], secure outsourced computation [4,5], and privacy-preserving neural networks [6,7].

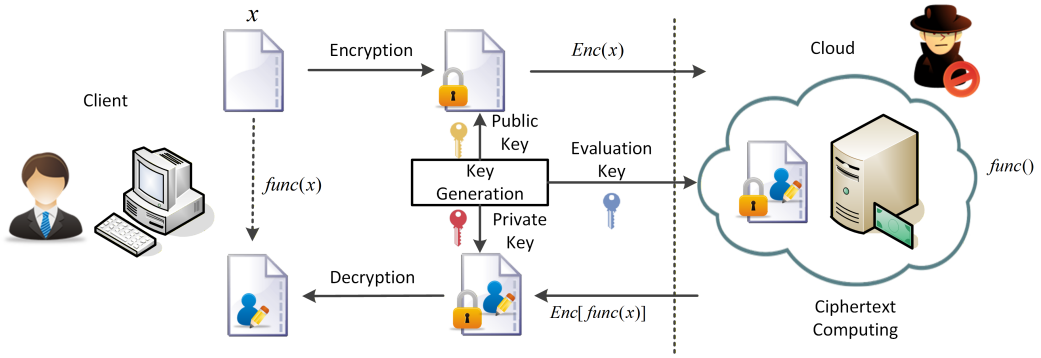


Figure 1. Cloud Computing Privacy Protection Model Based on FHE.

Since Gentry [8] proposed the first fully homomorphic encryption (FHE) scheme in 2009, FHE has entered a rapid development phase, with various schemes continuously emerging, performance being significantly improved, and application scope steadily expanding. However, computational efficiency remains a critical bottleneck limiting its practical deployment and further advancement. Even the most efficient FHE schemes to date, which are based on the Ring Learning with errors (RLWE) problem [9], suffer from low computational efficiency. This is mainly attributed to large polynomial dimensions ranging from thousands to tens of thousands, lengthy ciphertext coefficients with hundreds to thousands of bits, and the high computational complexity of the bootstrapping algorithm [10] required to support infinite-depth circuit evaluation. Accordingly, algorithm optimization and hardware acceleration for FHE schemes, as well as their underlying core arithmetic components, have become a prevailing research focus in the privacy computing community. To comprehensively clarify the current progress, technical limitations, and optimization trends of FHE hardware implementation, it is necessary to sort out existing relevant surveys and summarize state-of-the-art research.

Currently, a number of existing surveys [11–15] have systematically reviewed the theoretical framework, algorithm classification, and software implementation of FHE schemes. These works mainly focus on the mathematical principle, security attributes, and software library development of mainstream FHE schemes, including BGV [16], BFV [17,18], and CKKS [19]. Nevertheless, most of these surveys concentrate merely on software-level implementations and lack in-depth investigation and comprehensive comparison of FHE hardware acceleration designs. Several other surveys [20–23] preliminarily explore the software and hardware acceleration of FHE and summarize general acceleration methods based on CPU, GPU, FPGA, and ASIC platforms. However, the work in [20] only covers FHE acceleration studies from 2019 to 2022, failing to present the complete evolutionary trajectory of FHE hardware implementation. Moreover, the survey in [21] mainly emphasizes the overall accelerator architecture and software-hardware co-design strategies, while neglecting the hardware implementation status of core computational components. In addition, the survey [22] only focuses on the bootstrapping optimization technology of the CKKS scheme, without involving hardware acceleration for other FHE schemes and their core operation units. The work [23] conducts a review dedicated to FHE-specific hardware accelerators. However, it merely concludes the overall hardware framework of

existing accelerators and lacks in-depth analysis of fine-grained core modules, especially polynomial multipliers and modular multipliers.

In summary, existing literature either overemphasizes algorithm theory and software realization, or only briefly discusses coarse-grained accelerator architectures, lacking specialized analysis and horizontal comparison of key computing modules that dominate hardware performance. This motivates us to conduct a dedicated survey to comprehensively summarize the state-of-the-art of FHE hardware implementation from the perspectives of FHE scheme classification, overall accelerator architecture, and core component hardware design. This paper adopts a narrative review approach and provides comprehensive coverage of key research advances in the field of hardware acceleration for FHE within the targeted research period. To ensure a focused and reproducible survey, we searched major academic databases, including IEEE Xplore, ACM Digital Library, SpringerLink, ScienceDirect, Web of Science, Google Scholar, and IACR ePrint. The search covered publications from 2009 to 2026, with particular attention to recent studies after 2020. The main search terms included "fully homomorphic encryption", "FHE hardware implementation", "FHE accelerator", "FPGA/ASIC accelerator", "polynomial multiplier", "NTT/INTT", "Toom–Cook", "Karatsuba", and "modular multiplier". After removing duplicates, the retrieved papers were screened by title, abstract, and full text. We included works directly related to FHE schemes, hardware accelerator architectures, polynomial multiplication, and integer modular multiplication, while excluding papers focusing only on software libraries, application-level studies without hardware implementation, and unrelated cryptographic hardware. The remaining studies were organized according to FHE scheme evolution, accelerator type, core arithmetic component, implementation platform, which formed the basis of this survey. This work aims to fill the research gap in the field of FHE hardware-oriented systematic review and provide instructive references for the subsequent design and optimization of FHE hardware accelerators.

The contributions of this survey can be summarized as follows.

- We systematically sort out the research status of FHE schemes in chronological order, present the characteristics of typical schemes, and extract and analyze their common features and main performance bottlenecks.
- We provide a state-of-the-art review and characteristic analysis of the overall hardware implementations for FHE accelerators. In addition, we discuss the implementation characteristics of their core arithmetic components from the aspects of polynomial multipliers and integer modular multipliers.
- Finally, based on a comprehensive analysis of the hardware implementation architectures of major FHE accelerators, we propose the challenges currently faced in FHE accelerator design and the possible main development directions in the future.

The rest of this article is organized as follows. Section 2 reviews progress on FHE schemes, compares several representative schemes, and summarizes their key features. Section 3 outlines the development of FHE accelerators based on three hard problems and application-oriented FHE accelerator and highlights typical accelerator characteristics. Section 4 presents advances in accelerating core FHE components using three algorithm-based methods. Section 5 analyzes existing FHE accelerator architectures and suggests future research directions.

2. Overview of FHE Schemes

In this section, we review the research development progress of FHE schemes, discuss the main characteristics of FHE schemes proposed at different stages, and analyze and compare several typical FHE schemes in terms of core components, operation types, and parameter settings.

2.1. Research Progress of FHE Schemes

According to the timeline and algorithmic construction basis, FHE schemes are mainly divided into the first, second, and third generations. Specifically, in 2009, Gentry constructed the first FHE scheme (Gen09 [24]) that can support arbitrary homomorphic addition and multiplication simultaneously, based on hard problems over ideal lattices, marking the birth of the first-generation FHE schemes. The Gen09 scheme reduces the depth of the decryption circuit by compressing it and homomorphically evaluates the noisy result ciphertext through the bootstrapping algorithm. Relying on these two technologies, Gen09 can control the ciphertext noise within a limited range, thereby supporting unlimited homomorphic evaluation operations. Smart and Vercauteren further proposed the SV10 scheme [25] with smaller ciphertext and key sizes, while Stehlé and Steinfeld proposed the SS10 scheme [26] with a shallower decryption circuit depth. Subsequently, Gentry and Halevi improved the bootstrapping algorithm, proposing the GH11 [27] and GHS12a [10] schemes to further improve the efficiency of homomorphic decryption.

Discussion: From the perspective of algebraic properties, first-generation fully homomorphic encryption schemes provide a fundamental construction that supports an arbitrary number of homomorphic additions and multiplications, thereby establishing the theoretical feasibility of general-purpose encrypted computation. However, their algebraic structure is relatively complex and often requires costly bootstrapping procedures to control noise growth. In terms of security, their security mainly relies on the sparse subset sum problem, which cannot be strictly reduced to typical lattice hard problems, making the security foundation less standard than that of later lattice-based schemes. From an engineering perspective, these schemes suffer from high computational complexity, large ciphertext size, and low execution efficiency. Therefore, although they are of great theoretical significance, their practical deployment is greatly limited.

In 2011, Brakerski and Vaikuntanathan constructed the FHE schemes BV11a [28] and BV11b [29] based on the Ring Learning with Errors (RLWE) problem [9] and the Learning with Errors (LWE) problem [30], respectively. They reduced the security of these schemes to standard hard problems over ideal lattices, ushering in the era of second-generation FHE schemes. Thereafter, Brakerski, Gentry, and Vaikuntanathan constructed the leveled FHE scheme BGV [16] based on the BV11b scheme. This scheme can control the expansion of ciphertext dimension by performing KeySwitching after each homomorphic multiplication, and convert the exponential growth of ciphertext noise into linear growth through ModulusSwitching. In the same year, Gentry et al. developed the GHS12b scheme [31] with a larger plaintext space based on the BGV scheme, while Brakerski proposed the scale-invariant FHE scheme Bra12 [18]. Subsequently, Fan and Vercauteren further proposed the ring-based variant FV [17] of Bra12 (also referred to as the BFV scheme, which is uniformly used throughout this paper). This scheme introduced two optimized versions of the Relinearization technique to reduce the size of relinearization keys and enhance the efficiency of homomorphic evaluation. To avoid multi-precision operations in the BFV scheme and enable the Residue Number System (RNS) representation [32] to be compatible with the division and rounding operations of polynomial coefficients, Bajard et al. further improved the BFV scheme and proposed its RNS variant BEHZ [33], thereby effectively improving the computational efficiency of the RNS-BFV scheme. In 2019, aiming at the high time consumption of large integer division operations in the BEHZ scheme, Halevi et al. proposed the HPS scheme [34] based on the BEHZ scheme. This scheme adopted combined operations of integers and floating-point numbers, simplifying the implementation of the RNS-BFV scheme and accelerating its computational speed. Cheon et al. proposed the CKKS [19] FHE scheme for real-number computations based on the BGV scheme. This scheme abandoned the modulus of the plaintext space and reduced ciphertext noise

by Rescaling the computation results during homomorphic operations, while retaining the approximate value of the original plaintext with a predetermined precision. To address the limitation that the CKKS scheme could not support RNS decomposition and Number Theoretic Transform (NTT) optimization techniques, they further proposed the RNS variant of the CKKS scheme [35]. By introducing an RNS-compatible approximate modulus conversion technique, this variant effectively avoided RNS conversion and multi-precision operations. Subsequently, Han and Ki proposed a general variant of the CKKS scheme [36], which further reduced the temporary modulus for KeySwitching in the scheme and halved the computational complexity of homomorphic multiplication. At Eurocrypt 2024, Ma et al. [37] proposed a specific scheme that decomposes the Number Theoretic Transform (NTT) in the BGV bootstrapping process into multiple linear sub-transforms. These decomposed sub-transforms can efficiently complete homomorphic operations based on the fast NTT algorithm, significantly improving bootstrapping efficiency.

Discussion: From the perspective of algebraic properties, most second-generation FHE schemes support homomorphic operations over large plaintext spaces and are generally constructed over polynomial rings, which makes them well-suited for arithmetic homomorphic operations. By adopting optimization techniques such as batching [38], ciphertext packing [39], RNS technology [32], and Chinese Remainder Transform (CRT) [40], these schemes can process multiple plaintext slots in parallel and significantly improve execution efficiency. In terms of security, most second-generation FHE schemes are based on standard lattice hardness assumptions, such as LWE, RLWE, or their variants, providing a more solid and widely accepted security foundation than early constructions. From an engineering perspective, the BFV and CKKS schemes have gained extensive attention and applications in exact computation and approximate computation scenarios, respectively, due to their relatively mature software ecosystems and practical performance advantages. However, these schemes usually involve large ciphertext sizes, complex parameter selection, and high memory and computational overhead. Moreover, since most second-generation FHE schemes are leveled FHE schemes, complex bootstrapping algorithms are still required to support unlimited homomorphic evaluation operations. Therefore, second-generation FHE schemes achieve a relatively good balance among algebraic expressiveness, security foundation, and engineering practicality, but their bootstrapping cost and resource overhead remain important obstacles to fully practical deployment.

In 2013, Gentry, Sahai, and Waters constructed the GSW [41] FHE scheme based on the LWE problem using approximate eigenvectors, marking the entry of FHE into the third development phase. Both homomorphic addition and multiplication of this scheme are performed on ciphertext matrices; thus, there is no problem of ciphertext dimension expansion, and complex relinearization operations are no longer required. However, this scheme is still a leveled FHE scheme, and the ciphertext matrix consists of multiple LWE instances, resulting in a large expansion factor. Alperin and Peikert constructed an efficient bootstrappable FHE scheme AP14 [42] based on the GSW scheme by using a special outer scheme to homomorphically decrypt the original inner scheme. Ducas and Micciancio further constructed a more efficient bit-bootstrappable FHE scheme FHEW [43] based on the GSW and AP14 schemes. Relying on the GSW ciphertext form over polynomial rings and the most significant bit extraction function (msbExtract), this scheme can reduce the running time of the bootstrapping algorithm to less than 1 s. Chillotti et al. constructed the TFHE [44] (also known as CGGI16) FHE scheme with higher bootstrapping efficiency (less than 0.1 s) based on the characteristic that the multiplication result over the Torus ring is no larger than the multipliers. This scheme replaced the inner product operation between TGSW and TGSW over the original Torus ring with the outer product operation between TGSW and TLWE, thereby reducing the bootstrapping time of TFHE to 52 millisec-

onds. This time was further reduced to 13 milliseconds by optimizing the homomorphic accumulator [45] In 2017. Given that TFHE only supports binary keys, making it vulnerable to hybrid attacks [46], and the operations over the Torus ring cannot be directly compared in performance with the integer modular operations adopted by other schemes, Micciancio and Polyakov further improved and proposed the FHEW-like [47] FHE framework in 2021, which can support both FHEW and TFHE schemes over polynomial rings. By optimizing the initialization operation, this framework avoids the complex msbExtract operation in the original FHEW scheme. Meanwhile, it further improves the execution efficiency of the homomorphic accumulator and bootstrapping algorithm by replacing the multiplication operation between two RGSW ciphertexts with that between one RGSW ciphertext and one RLWE ciphertext. Bae et al. [48] proposed a new bootstrapping mode—METABTS—to address the problem that modulus switching in the bootstrapping process introduces errors. Essentially, this method is a type of bit extraction algorithm; its core idea is to double the precision of the bootstrapping result by repeatedly executing any given basic bootstrapping algorithm twice. Based on GSW variant schemes constructed upon the Nth-degree Truncated Polynomial Ring Unit (NTRU) problem, Bonte et al. [49] proposed the FINAL scheme with higher efficiency than the TFHE scheme. This scheme focuses on the integration and optimization of NTRU and LWE problems. It achieves NTRU-to-LWE ciphertext conversion by leveraging the existing KeySwitching procedure, thereby breaking the security and bootstrapping efficiency bottlenecks of conventional FHE schemes. Lee et al. [50] proposed a number of bootstrapping optimization techniques for the TFHE scheme, which effectively reduced the time overhead of the TFHE bootstrapping process and the size of the bootstrapping key while ensuring the same security level. Xiang et al. [51] designed a GSW-like encryption scheme based on the NTRU assumption, which can realize fast KeySwitching of ring automorphism. Relying on ring automorphism and KeySwitching technology, a new blind rotation algorithm and bootstrapping algorithm were proposed. At Eurocrypt 2024, Wang et al. [52] redesigned the workflow of circuit bootstrapping, improved the LMKC+ blind rotation algorithm [53], and proposed a new circuit bootstrapping algorithm, whose performance is superior to the existing TFHE methods. In 2025, Zhao et al. [54] introduced a block binary distribution in the key generation process of the encryption scheme, so that each block of the key only needs to perform one outer product operation, thereby reducing the total number of outer products required by the blind rotation algorithm.

Discussion: From the perspective of algebraic properties, third-generation FHE schemes mainly focus on the optimization and improvement of bit-bootstrappable algorithms for single-bit plaintexts, and are particularly suitable for logical homomorphic evaluation operations. Their algebraic design enables efficient gate-level computation, but the support for large-integer or approximate arithmetic is usually less direct than that of schemes based on polynomial rings. In terms of security, most third-generation schemes are built on lattice-based assumptions such as LWE or its variants, providing a more standard and widely studied security foundation. From an engineering perspective, they feature high bootstrapping efficiency, relatively simple Boolean circuit evaluation, and strong applicability to privacy-preserving logic computation. However, when applied to arithmetic-intensive tasks, the bit-level representation may introduce additional circuit depth and conversion overhead. Overall, these schemes have become one of the key research areas in current FHE scheme research.

2.2. Analysis of Typical FHE Schemes

The main existing FHE schemes are mainly divided into two categories, the second-generation leveled schemes represented by BFV and CKKS, and the third-generation bit-bootstrappable schemes represented by FHEW/TFHE. These two categories have sig-

nificant differences in core components, processing technologies, core operations, cryptographic parameters, and other aspects, while sharing a common underlying operational basis. These characteristics not only determine the performance bottlenecks of homomorphic evaluation but also lay an algorithmic foundation for FHE hardware design. The main characteristics are summarized in Table 1.

Table 1. Characteristics of main FHE schemes.

Scheme	BFV	CKKS	FHEW/TFHE
Type	Leveled FHE	Leveled FHE	Bit-bootstrappable
Core component	Homomorphic multiplication, including relinearization	Homomorphic multiplication, including relinearization	Homomorphic accumulator
Parallel technology	RNS	RNS	None
Modulus conversion technology	Base extension/scaling	Approximate up/down modulus	None
Core operation	Polynomial multiplication	Polynomial multiplication	Polynomial matrix multiplication
Basic operation	Integer multiplication/modular operation	Modular operation	Modular operation
Dimension (N)	$\geq 2^{12}$	$\geq 2^{12}$	2^{10} or 2^{11}
Modulus ($\log_2 Q$)	$\geq$100	$\geq$100	$\leq$50
Modulus selection	Prime	Prime	Power of 2 or prime

- In terms of core components and processing techniques, the core component of leveled FHE schemes represented by BFV and CKKS is homomorphic multiplication (including relinearization), which accounts for the vast majority of time overhead in homomorphic evaluation. To avoid multi-precision computation caused by large moduli, both schemes adopt the Residue Number System (RNS) for parallel processing, decomposing ciphertexts under large moduli into multiple ciphertexts under small moduli, thereby improving parallel performance in homomorphic evaluation. The difference is that the BFV scheme uses base expansion and base scaling to implement ciphertext multiplication and scaling-rounding operations in a larger modulus space, whereas the CKKS scheme employs approximate modulus upscaling and downscaling to perform relinearization in an extended modulus domain. In contrast, for bit-bootstrappable FHE schemes such as FHEW/TFHE, the core operation is the update of the homomorphic accumulator. Since their ciphertext moduli are much smaller than those of leveled FHE schemes and no modulus decomposition is involved, neither RNS parallel processing nor modulus conversion techniques are required.
- Regarding core operations and basic arithmetic types, the central computation of leveled FHE schemes (BFV, CKKS) is polynomial multiplication. Specifically, the BFV scheme involves integer multiplication, modular multiplication, modular addition, and modular subtraction. Integer multiplication is mainly used in base expansion and scaling, while integer modular operations run throughout the entire homomorphic evaluation process. Although the CKKS scheme involves approximate modulus upscaling and downscaling, it still relies primarily on modular multiplication, addition, and subtraction over integers. For bit-bootstrappable schemes represented by FHEW/TFHE, the core operation is mainly multiplication between polynomial vectors and polynomial matrices. Similarly, the FHEW/TFHE scheme only involves basic modular arithmetic operations including modular multiplication, addition, and subtraction.
- In terms of the size and type of cryptographic parameters, leveled FHE schemes (BFV, CKKS) use a polynomial dimension of at least 2^{12}, which typically increases to 2^{16}–2^{17} to support bootstrapping. The ciphertext modulus is generally larger than 100 bits and can even reach hundreds to thousands of bits for deeper multiplicative circuits. Since leveled FHE schemes use RNS to control the modulus size on each RNS branch, the ciphertext moduli are usually chosen as prime numbers. By comparison, bit-bootstrappable FHE schemes such as FHEW/TFHE employ much smaller polynomial

dimensions and ciphertext moduli while the polynomial dimension is typically set to 2^{10} or 2^{11}, and the ciphertext modulus is usually below 50 bits (between 20 and 40 bits). The modulus for FHEW can be either a power of two or a prime. Since the initialization step of FHEW/TFHE requires assigning specific values to polynomial coefficients $Q/8$, using a power-of-two modulus facilitates precise coefficient assignment $Q/8$ and simplifies modular reduction to simple bit-shift operations.

Discussion: Based on the above analysis, the main performance bottleneck of FHE schemes lies in homomorphic evaluation operations. Among them, homomorphic evaluation, especially homomorphic multiplication, involves a large number of polynomial multiplications with large polynomial dimensions and ciphertext moduli. Therefore, in addition to the overall hardware acceleration for FHE schemes, hardware acceleration for polynomial multiplication has become the top priority. For leveled FHE schemes represented by BFV and CKKS, ciphertext moduli are usually selected as prime numbers to support RNS parallel processing. In contrast, bit-bootstrappable FHE schemes such as FHEW/TFHE adopt relatively small polynomial dimensions and ciphertext moduli, where the modulus can be either a power of two or a prime number. Such algorithmic characteristics provide a basis for the selection of polynomial multiplication hardware acceleration schemes. Furthermore, both leveled and bit-bootstrappable FHE schemes involve polynomial multiplication, integer modular multiplication, modular addition, modular subtraction and other operations, sharing a common underlying arithmetic foundation. On the other hand, different application scenarios and security requirements impose distinct constraints on the type of encryption scheme, the type and dimension of polynomial multiplication, the type and size of ciphertext moduli, and the decomposition base size, which demands high algorithmic flexibility for FHE hardware implementations. All these features lay an algorithmic foundation for the design of FHE hardware accelerators.

Accordingly, subsequent chapters conduct a review and analysis of FHE hardware acceleration implementations from two dimensions including overall hardware acceleration and core component acceleration. The detailed classification framework of relevant hardware architectures is presented in Figure 2.

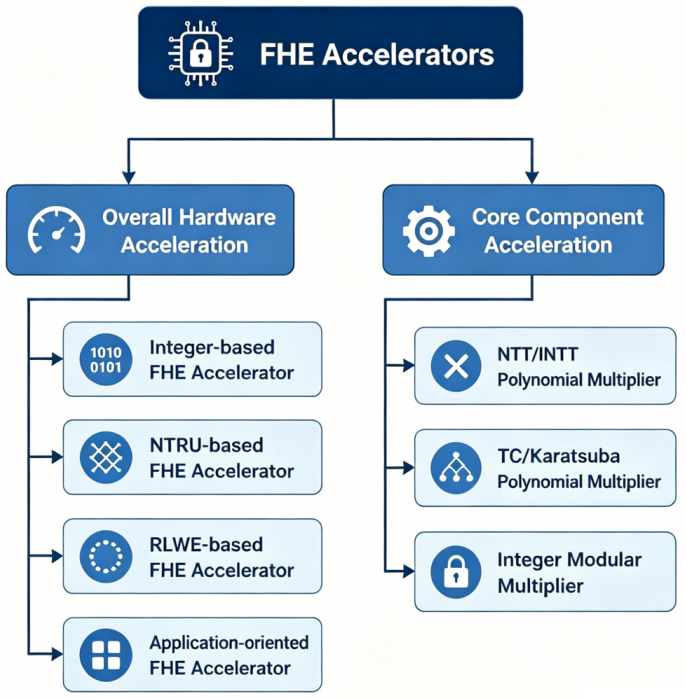


Figure 2. Classification Framework of FHE Hardware Architectures.

3. Overall Hardware Acceleration

According to the types of supported FHE schemes, the implementations of FHE accelerators mainly fall into four categories: integer-based, NTRU-based, RLWE-based and Application-oriented FHE accelerator. The main types and characteristics of the first three categories of FHE accelerators are summarized in Table 2.

Table 2. Summary of FHE hardware accelerators.

Type	Authors	Architecture	Platform	Scheme	Strength	Weakness
Integer	Wang [55]	N/A	FPGA (Stratix-V)	GH	First work integrating SSA with FFT for GH-FHE.	Customized for 768K-bit multiplication, lacks generality.
	Doroz [56]	N/A	ASIC (90 nm TSMC)	GH	First full hardware acceleration of GH-FHE.	High hardware cost, limited generality.
	Moore [57]	N/A	FPGA (Virtex 7)	Coron	Efficient use of DSP slices with lightweight design.	DSP-dependent, limited flexibility.
NTRU	Pöppelmann [58]	N/A	FPGA (Stratix-V)	YASHE	First complete hardware acceleration for YASHE.	Highly specialized, poor generality.
	Roy [59]	N/A	FPGA (Virtex-7)	YASHE	Modular and reconfigurable design.	Extra control overhead.
	Öztürk [60]	N/A	FPGA (Virtex 7)	LTV-SHE	First FPGA accelerator for LTV-SHE.	Scheme-dependent, limited reuse.
	Cousins [61]	HEPU	FPGA (Virtex-7)	LTV	Efficient CRT/iCRT acceleration.	Scheduling overhead.
RLWE	Roy [62]	HEPCloud	CPU + FPGA (Virtex 6)	BFV	First multi-core BFV accelerator.	Poor compatibility.
	Mert [63]	N/A	CPU + FPGA + ASIC	BFV	Lightweight NTT optimization.	Limited reconfigurability.
	Turan [64]	HEAWS	FPGA (Virtex UltraScale+)	BFV	First AWS FPGA deployment.	Poor generality.
	Riazi [65]	HEAX	CPU + FPGA + ASIC (Stratix10)	CKKS	Multi-layer parallelism.	High resource overhead.
	Roy [66]	Medha	FPGA (Alveo U250)	CKKS	Multi-core + instruction-level parallelism.	Limited to RNS-CKKS.
	Samardzic [67]	F1	ASIC (14/12 nm)	BGV/CKKS	High algorithm generality.	Large ASIC area.
	Samardzic [68]	CraterLake	ASIC (14/12 nm)	BGV/CKKS	Supports unbounded FHE.	High complexity.
	Mert [69]	Media opt.	FPGA + ASIC (Alveo U250/7 nm)	CKKS	Microcoded instruction-set design.	Limited compatibility.
	Kim [70]	ARK	ASIC (7 nm)	BGV/CKKS	High memory efficiency.	Complex communication.
	Yang [71]	Poseidon	FPGA (Alveo U280)	CKKS	Full bootstrapping support.	Complex control.
	Bertels [72]	N/A	FPGA (ZCU102)	FHEW	Fastest FHEW implementation.	External key storage.
	Jiang [73]	MATCHA	ASIC/GPU (16 nm PTM/Tesla V100)	TFHE	FFT without multiplication.	Limited compatibility.
	Van [74]	FPT	FPGA (Alveo U280)	TFHE	First fixed-point FFT for TFHE.	External key dependency.
	Zhang [75]	N/A	FPGA (Alveo U280)	CKKS/TFHE	Cross-scheme switching.	One-way only.

3.1. Integer-Based FHE Accelerators

In 2013, Wang et al. [55] designed an FPGA-based hardware accelerator for the GH scheme [76]. To accelerate large-integer multiplication in the scheme, the accelerator employed the Schönhage–Strassen algorithm [77] to construct a 64K-point FFT acceleration engine, and adopted Solinas moduli for modular reduction optimization. At 100 MHz, the FPGA implementation is about twice as fast as the same FFT algorithm executed on the NVIDIA C2050 GPU which has 448 cores running at 1.15 GHz but at much lower power consumption. Doröz et al. [56] developed the first ASIC-based FHE accelerator for the GH scheme using a 90 nm process. This accelerator supported encryption, decryption, and recryption primitives. To speed up large-integer multiplication, they built an algorithm- and cache-optimized million-bit large-integer multiplier based on the Schönhage–Strassen algorithm. By utilizing a number of optimizations, including reformulation of the operations, use of spectral techniques and precomputation to speed up the arithmetic operations, the presented architecture achieves to realize the encryption, decryption, and recryption operations, in 18.1 ms, 16.1 ms, and 3.1 s, respectively, and occupies a footprint of less than 30 million gates. Moore et al. [57] proposed an accelerator architecture for the Coron scheme [78]. Constrained by the bit-width of DSP48E1 slices, they presented a structurally optimized Comba multiplier [79] hardware architecture, which effectively improved the

performance of large-integer multiplication. Subsequently, they further proposed a compact accelerator architecture and a multi-core parallel accelerator architecture for the Coron scheme [80]. By adopting the Comba multiplication scheduling scheme to handle large-integer multiplications with multiplicands of unequal bit lengths, they drastically reduce the number of memory read and write operations in hardware. Under medium-sized security parameters, the encryption operations of two architectures achieve speedup factors of 3.6 and 10.4 respectively.

Discussion: Existing integer-based FHE accelerators show a highly homogeneous optimization pattern, mainly focusing on accelerating large-integer multiplication through FFT-based methods (e.g., Schönhage–Strassen) or Comba-style multipliers, with performance improvements driven by parallelization, cache optimization, and hardware-level circuit restructuring rather than fundamental changes at the algorithmic or cryptographic level. While this multiplication-centric design is effective under moderate parameter settings, it becomes increasingly inefficient as FHE security requirements grow, particularly with continuously expanding key sizes that lead to exponential increases in hardware complexity and memory pressure. As a result, the key limitation is not simply performance, but the lack of scalability and algorithmic adaptability, encompassing FFT-based approaches suffer from severe latency and implementation complexity under ultra-large word sizes, while Comba-based designs are constrained by sequential dependencies and memory bottlenecks. Moreover, evolving FHE schemes and security assumptions further exacerbate compatibility and potential security risks. These factors together explain why, despite early gains in schemes such as GH and Coron, research on integer-based FHE accelerators has become relatively scarce in recent years due to overlarge key parameters, limited efficiency improvements, and inherent vulnerabilities in underlying schemes.

3.2. NTRU-Based FHE Accelerators

In 2015, Pöppelmann et al. [58] presented an FPGA-based hardware implementation for the YASHE scheme [81]. To improve the computational throughput of polynomial multiplication, they adopted a double-buffered Cache-NTT algorithm and optimized the memory access mechanism. For the parameter set ($n = 16384$, $\lceil \log_2 q \rceil = 512$) capable of evaluating 9 levels of multiplications, they can perform a homomorphic addition in 0.94 ms and a homomorphic multiplication in 48.67 ms. In the same year, Roy et al. [59] proposed a multi-core parallel accelerator architecture for the YASHE scheme. This design adopts a multi-accelerator and multi-core parallel framework and integrates multiple optimization strategies, including CRT- and NTT-based polynomial multiplication units with optimized memory access, a fast Barrett-like polynomial reduction method, and efficient divide-and-round and CRT units for ciphertext multiplication. Such improvements significantly boost the ciphertext computation efficiency [82]. As a result, the implementation finishes SIMON-64/128 computation in 171.3 s by processing 2048 ciphertexts in parallel, yielding a per-block latency of 83.6 ms and achieving a 24.5× speedup over the state-of-the-art software implementation. Öztürk et al. [60] designed a heterogeneous CPU + FPGA accelerator platform for the LTV scheme [83]. Implemented on the Virtex-7 FPGA, this accelerator supports ciphertext computations including CRT transformation, modulus switching, relinearization, and the AES algorithm [84], and achieves 442 ms per AES block homomorphic evaluation, outperforming CPU/GPU counterparts by 28.5× and 17×, respectively. Later, they further refined the accelerator design [85], extended the polynomial dimension, and added support for ciphertext computation of the Prince block cipher [86]. Cousins et al. [61] developed a homomorphic encryption co-processor architecture called HEPU for the LTV scheme. To alleviate computational bottlenecks, HEPU used pipelining

to accelerate NTT-based polynomial multiplication and implemented hardware-accelerated CRT and inverse CRT (ICRT) transformations.

Discussion: Existing NTRU-based FHE accelerators, primarily targeting schemes such as YASHE and LTV, demonstrate substantial performance gains by accelerating core polynomial operations, especially NTT-based multiplication, CRT transformations, and modulus switching through FPGA-friendly architectures and heterogeneous CPU-FPGA designs. However, despite these advancements, the overall research trajectory remains relatively limited. This is largely due to inherent subfield attack vulnerabilities [87] in NTRU-based constructions, which undermine long-term cryptographic robustness and restrict their adoption in high-security FHE settings. From a broader perspective, similar to integer-based accelerators, NTRU-based designs are also dominated by multiplication-centric optimization strategies at the polynomial arithmetic level, with limited exploration of algorithmic or scheme-level innovation. Consequently, although both integer-based and NTRU-based FHE accelerators achieve notable performance improvements in specific regimes, their development is constrained by scalability concerns, memory and parallelization bottlenecks.

3.3. RLWE-Based FHE Accelerators

In 2018, Roy et al. [62] proposed HEPCloud, the first hardware accelerator architecture tailored for the BFV scheme [17,18]. To support the RNS-BFV homomorphic evaluation operations in the BEHZ scheme [33], the accelerator was designed with a unified hardware structure for $Lift_{q\to Q}$ and $Lift_{Q\to q}$ functions, and the Number Theoretic Transform (NTT)-based polynomial multiplication was optimized. However, the homomorphic evaluation performance of the accelerator was constrained by the low memory access speed. They further developed a domain-specific multi-core accelerator architecture for the BFV scheme [88]. Specifically, the FPGA end integrated two co-accelerators to accelerate homomorphic evaluation operations, with each co-accelerator comprising seven polynomial arithmetic units and one Lift-and-Scaling unit. By designing an efficient dual-core NTT processing structure and optimizing the memory access strategy, their implementation achieves over 13x speedup with respect to a highly optimized software implementation. In the same year, Mert et al. [63] designed iterative and four-step NTT hardware architectures to accelerate the encryption and decryption algorithms of the BFV scheme. The SEAL library [89] was adopted on the CPU end to generate key parameters, while the encryption and decryption operations were offloaded to the FPGA end for acceleration, enabling the accelerator to achieve almost 12× and 7× latency speedups compared to pure software implementations. Turan et al. [64] implemented the first acceleration of the RNS-BFV scheme based on Amazon's AWS FPGA development board. To accelerate computations across different RNS channels, the accelerator integrated seven co-accelerators, each of which further contained multiple dedicated cores. Through optimizations of the accelerator architecture and the NTT memory access strategy, the accelerator achieved a homomorphic multiplication evaluation rate of 613 operations per second. Riazi et al. [65] proposed HEAX, the first heterogeneous acceleration platform based on CPU + FPGA for the CKKS scheme [19]. To improve the memory access efficiency of NTT, HEAX optimized the multi-core memory access pattern and pipelining strategy of NTT. By adopting a multi-core and pipelined parallel processing structure for KeySwitching, which is the most time-consuming operation, HEAX achieved 164–268× performance improvement for a wide range of FHE parameters. Nevertheless, the fully unrolled multi-core processing structure also resulted in increased hardware resource overhead. Roy et al. [66] proposed Medha, a programmable instruction set architecture (ISA) for the RNS-CKKS scheme. By adopting a multi-core accelerator hardware architecture and an instruction-level parallel processing strategy,

Medha effectively improved the performance of RNS-CKKS homomorphic evaluation while eliminating the need for off-chip memory access. On a Xilinx Alveo U250 card, it reduced latency by 2.4× and occupied a smaller area than state-of-the-art reconfigurable accelerators. In the same year, Samardzic et al. [67] presented F1, the first programmable FHE accelerator. This accelerator adopted a multi-cluster parallel accelerator architecture, where each cluster included multiple processing elements (PEs) with different functions. By optimizing each PE unit and adopting a leveled memory access mechanism to decouple data movement, F1 could efficiently support homomorphic evaluation operations of both BGV and CKKS schemes, and outperform state-of-the-art software implementations by gmean 5400× and by up to 17,000×. In 2022, Mert et al. [69] further optimized the Medha accelerator architecture [66] and adopted a divide-and-conquer strategy, enabling homomorphic evaluation operations over polynomial rings $R_{Q,2N} = \mathbb{Z}_Q[x]/(x^N + 1)$, to be decomposed into ring-level operations $R_{Q,N} = \mathbb{Z}_Q[x]/(x^{2^N} + 1)$. This design allowed flexible support for different fully homomorphic cryptographic parameters. For the large parameter sets $(\log Q, N) = (438, 2^{14}), (546, 2^{15})$, it achieved accelerations by up to 68× and 78× times respectively compared to a highly optimized software implementation. However, the decomposition and merging operations introduced additional hardware resource and latency overhead. In the same year, Kim et al. [70] proposed ARK, a FHE acceleration architecture for the CKKS scheme. This architecture eliminated the off-chip memory bandwidth bottleneck through runtime data generation and interoperable key reuse, and simplified dataflow organization through an efficient interleaved data allocation strategy and customized functional units, thereby minimizing on-chip data movement. Compared with the single-threaded software implementation on CPU, ARK achieved 3139×, 104×, and 18× performance speedup compared to Lattigo, 100×, and F1 accelerator [67], respectively. In 2023, Yang et al. [71] proposed Poseidon, a practical fully homomorphic cryptographic accelerator for the CKKS scheme, which could support complex FHE operations such as bootstrapping, key switching, and rotation under limited FPGA resources. By optimizing modular addition, modular multiplication, NTT, automorphism and shared Barrett reduction unit, it achieved up to 10.6×/8.7× speedup over GPU and the ASIC solution for the FHE benchmark. In the same year, Lee et al. [90] proposed a configurable and efficient encryption/decryption accelerator hardware architecture suitable for the CKKS scheme, which could flexibly support various polynomial scales. The average encryption speed of this architecture was 23.7× higher than that of the SEAL homomorphic encryption library. In 2024, Bae et al. [91] proposed HERMES, a novel efficient ring packing method that used MLWE-format ciphertexts to reduce memory consumption. In a single-threaded implementation, packing multiple LWE ciphertexts into one RLWE ciphertext took 10.2 s, and the throughput was 41× higher than that of PEGASUS, the previous state-of-the-art scheme. Next year, Soni et al. [92] designed B512, a vector instruction set architecture (ISA) customized for ring processing requirements, and the corresponding Ring Processing Unit (RPU). Under the GF 12 nm process, the RPU unit achieved NTT acceleration that is three orders of magnitude faster than that of the CPU.

Bertels [72] proposed an NTT-based accelerator architecture tailored for the FHEW scheme [43]. By integrating 160 parallel NTT units and equipping a fast memory interface, the accelerator effectively enhanced the bootstrapping efficiency of the FHEW scheme, enabling the completion of bootstrapping in less than 20 ms. However, this design imposed substantial hardware resource overhead, which limited its practical deployment potential. Samardzic et al. [68] developed CraterLake, an FHE accelerator built on the F1 accelerator architecture [67] that supports unbounded computation depth. It adopts a novel ultra-wide 2048-lane vector uniprocessor design distributing vector operations across the entire chip, distinct from earlier vector multicore architectures. This uniprocessor scheme cuts con-

current operations to shrink hardware footprint, lower off-chip data traffic, and simplify compiler design. The accelerator maintains wide compatibility with BGV, GSW, and CKKS schemes, attaining an 11.2× speedup over the original F1 accelerator. Jiang et al. [73] presented MATCHA, the first high-speed, energy-efficient ASIC accelerator dedicated to the TFHE scheme. It integrates a pipelined datapath with TGSW clusters and external product cores to enable intensive bootstrapping key unrolling and cut FFT/IFFT invocations in bootstrapping; separate register banks are deployed to accommodate sequential memory accesses for TGSW operations and irregular memory accesses for FFT/IFFT computations. Compared with the GPU-based homomorphic computation library cuHE [93], MATCHA achieves a 2.3× improvement in TFHE throughput. Van et al. [74] proposed FPT, a fixed-point FPGA accelerator designed specifically for TFHE bootstrapping. To accelerate the TFHE bootstrapping process, the FPT accelerator adopts approximate fixed-point arithmetic instead of traditional double-precision or single-precision floating-point operations. Leveraging its streaming architecture, FPT achieves a bootstrapping throughput improvement of two to three orders of magnitude compared with existing CPU-based implementations. In 2024, Zhang et al. [75] designed a dedicated hardware architecture to facilitate scheme switching between CKKS and FHEW. The computational module of this architecture supports coefficient conversion, modulus switching, and key switching operations, while the hardware designs for preprocessing and postprocessing stages are optimized to minimize latency. Compared with software-based baselines, the proposed hardware architecture improved the dynamic switching efficiency 365× on the Xilinx U280 Datacenter Acceleration FPGA.

Discussion: RLWE-based FHE accelerators have become the dominant direction due to stronger security foundations and more flexible algebraic structures. Recent works across BFV, BGV, CKKS, TFHE, and FHEW schemes show a clear shift from pure arithmetic optimization toward system-level designs, including multi-core and heterogeneous CPU-FPGA architectures, programmable ISAs (e.g., Medha, F1), deep pipelining, and memory-aware dataflow optimization. These innovations enable substantial speedups ranging from tens to thousands of times over software implementations while improving cross-scheme adaptability. Overall, RLWE-based designs exhibit better scalability and architectural flexibility, and these advantages make RLWE-based accelerators more suitable for practical deployment in diverse application scenarios.

3.4. Application-Oriented FHE Accelerators

To tackle the severe computational overhead of FHE in practical privacy-preserving scenarios, application-oriented FHE accelerators have emerged as a critical research direction. They adopt targeted hardware-algorithm co-optimization adapted to diverse task features and deployment constraints. Current mainstream research falls into three application-oriented branches including edge computing, federated learning, and artificial intelligence. This section systematically reviews the research progress, technical features and development trends of FHE accelerators for the three scenarios.

(1) Edge computing oriented FHE accelerator. FHE accelerators for edge computing support privacy-preserving edge tasks including intelligent inference, edge-cloud collaborative analysis, federated secure aggregation and sensitive data processing. Restricted by edge devices' limited computing, storage, power and bandwidth, pure software FHE cannot be deployed directly. Current research shifts from cloud ciphertext computation optimization to reducing client-side encryption, encoding, preprocessing and transmission overhead.

In terms of lightweight edge deployment, Chan et al. [94] presented HHEML, a hybrid HE hardware accelerator integrating lightweight symmetric encryption and FHE. It lets edge devices execute low-cost encryption, while heavy homomorphic computations are offloaded to servers. Built around FHE-friendly lightweight symmetric ciphers, the design was verified on PYNQ-Z2 with MNIST inference, cutting client encryption latency by over 50× and boosting hardware throughput by nearly 2×. Aikata et al. [95] designed Pasta on Edge, a hybrid HE crypto processor that adopts symmetric encryption and homomorphic decryption to ease the computation and communication overhead of traditional FHE client public-key encryption. Implemented on 7 nm and 28 nm high-end ASICs, it delivers a 97× speedup against existing FHE public-key client accelerators. For core operator acceleration, Di Matteo et al. [96] developed an NTT accelerator for embedded Microsoft SEAL/CKKS. It optimizes CKKS operations via configurable NTT units, low-latency I/O memory and a twiddle factor generator, and was validated on an FPGA with a 32-bit RISC-V core. Hardware acceleration achieves roughly 1000× speedup compared with pure software execution. Guerrini et al. [97] put forward a resource-saving CKKS NTT FPGA accelerator deployed on Kria KV260 SoC and compatible with Full-RNS HEAAN. It supports $\log N \in [12, 16]$ and moduli up to 64 bits, lifting slice-equivalent efficiency by 20.2× over the fastest open-source accelerator and 1.98× over the most resource-efficient counterpart. Regarding complete client-side pipeline acceleration, Krieger et al. [98] proposed a compact unified CKKS hardware architecture covering encoding, encryption, decoding and decryption. Custom floating-point units, hardware-optimized floating-point modular reduction and shared resource reuse mitigate computation load under tight area constraints, with up to 59× speedup over previous hardware designs. These works illustrate that edge FHE accelerators evolve from separate NTT/modular multiplication acceleration to full client encryption pipeline software-hardware co-optimization.

Discussion: Edge computing FHE accelerators are forming a system-level paradigm "lightweight end-device encryption-homomorphic computation on cloud/edge servers-hardware-software collaborative optimization". Core design targets include low area, low power consumption, small storage footprint, parameter configurability and friendly device-cloud interaction. RISC-V SoCs, FPGA/multi-core SoCs and lightweight ASICs are all viable deployment carriers. Among them, NTT/INTT, modular multiplication and reduction, twiddle factor generation, CKKS encoding and encryption, as well as hybrid HE symmetric encryption kernels deliver the highest hardware acceleration gains.

(2) Federated learning oriented FHE accelerator. FHE accelerators tailored for federated learning focus on secure model aggregation. Clients encrypt local gradients or model parameters, and servers perform global aggregation without accessing plaintext updates, mitigating privacy risks including model inversion, attribute inference and untrusted server threats. Unlike edge inference, federated learning suffers bottlenecks beyond single encryption or homomorphic operation latency, including client-side encryption costs across multiple training rounds, ciphertext expansion, communication overhead, aggregation parallelism and on-chip data movement. Thus, research in this domain requires cross-layer co-optimization covering cryptographic operators, model parameter organization, communication compression and hardware dataflow.

Meng and Lyu [99] proposed FedBit, a BFV-based FHE acceleration scheme for federated learning with a hardware/software co-design framework. It adopts bit-interleaved data packing to embed multiple model parameters into a single ciphertext coefficient, alleviating ciphertext expansion and enhancing computational parallelism. Meanwhile, it integrates a dedicated FPGA accelerator for cryptographic operations and optimizes dataflow to reduce storage overhead. Experimental results demonstrate that FedBit delivers a two-order-of-magnitude speedup in encryption and reduces average communication

overhead by 60.7%. Yang et al. [100] developed an FPGA-based homomorphic encryption acceleration framework for FL training. Built on the Paillier cryptosystem, the framework adopts HLS-based hardware acceleration to optimize the processing cycles, resource usage and operating frequency of modular multiplication, reducing FL training encryption time by up to 71%. Though this work leverages partially homomorphic encryption instead of full FHE, its hardware acceleration design for the federated training pipeline provides valuable engineering references for subsequent FHE-FL system development. Chen et al. [101] presented SAFE, a scalable FHE accelerator architecture targeting the homomorphic matrix–vector multiplication bottleneck in vertical federated learning. Adopting algorithm–hardware co-design, the architecture supports coefficient-wise encoding, compression and concatenation modes, as well as HMVP dataflow-oriented parallel hardware, realizing end-to-end acceleration for federated logistic regression training on a CPU–FPGA heterogeneous system. Notably, SAFE extends FHE acceleration from parameter aggregation in horizontal federated learning to feature interaction, matrix computation and cross-institutional data collaboration in vertical federated learning.

Discussion: FHE accelerators for federated learning exhibit three trends. First, the optimization target has expanded from individual encryption/decryption processes to secure aggregation and the entire federated training pipeline. Second, the optimization methodology has shifted from single cryptographic operator acceleration to coordinated quantization, packing, communication compression, and dataflow co-design. Third, the application scenarios have expanded from model update aggregation in horizontal federated learning to homomorphic matrix operations and cross-institution feature collaboration in vertical federated learning. Therefore, the core challenge in this direction is not merely improving hardware throughput, but achieving a system-level balance among privacy protection strength, model accuracy, communication cost, and training efficiency.

(3) AI oriented FHE accelerator. AI oriented FHE accelerators mainly support privacy-preserving machine learning and encrypted neural network inference, accelerating computationally expensive operations including matrix–vector multiplication, convolution, nonlinear approximation, bootstrapping, key switching, and NTT/INTT. Different from general-purpose FHE accelerators, AI-oriented designs jointly optimize cryptographic primitives and neural network computation characteristics such as matrix operations, convolution structures, batching, ciphertext packing and data movement. Existing research can be classified into three categories: FPGA-based application-oriented acceleration, coarse-grained-reconfigurable-architecture (CGRA) based dataflow acceleration, and ASIC/SoC-based dedicated acceleration architectures.

Ren et al. [102] proposed CHAM, a customized accelerator for homomorphic matrix–vector products targeting privacy-preserving learning workloads including logistic regression and convolution operations. Adopting algorithm–hardware co-design with customized datapaths, parallel modular arithmetic units and HMVP-oriented dataflow, CHAM is implemented on Xilinx FPGA and adaptable to 2D and 3D convolution tasks. It achieves 1800× speedup for matrix–vector multiplication, 36× for logistic regression, and 144× for Beaver triple generation against prior works, verifying the efficacy of FPGA customization for linear-algebra-heavy encrypted AI workloads. Jiang et al. [103] presented a CGRA-based architecture for FHE workload acceleration, especially FHE-CNN inference. By mapping modular arithmetic, NTT, ciphertext rotation and polynomial operations to reconfigurable dataflow, the design improves operator reusability and kernel adaptability. Experimental results show it delivers 8.15× higher performance than conventional CGRA baselines and 16.48× better energy efficiency than state-of-the-art FPGA-based FHE-CNN accelerators. Reagen et al. [104] developed Cheetah, which integrates HE algorithm optimization and dedicated hardware to enable efficient private DNN inference.

HE parameter tuning and operator scheduling bring up to 79× speedup over previous schemes. The customized accelerator further optimizes encrypted DNN linear layers, convolution, data movement and task scheduling. Evaluated on ResNet50, VGG16 and AlexNet, the hardware achieves near-plaintext inference performance, shifting FHE acceleration research from isolated operator optimization to end-to-end private neural network inference optimization. Golder et al. [105] implemented HERACLES, a highly parallel FHE SoC fabricated with Intel 3 nm CMOS technology. Equipped with an 8192-way SIMD programmable architecture, high-throughput polynomial arithmetic datapaths and multi-level memory systems, it yields 1074×–5547× performance improvement in FHE mathematical operations compared with a 24-core Intel Xeon W7-3455 processor. Though not tailored for specific neural network models, it provides a solid hardware foundation for future large-scale encrypted AI inference.

Discussion: Existing studies demonstrate that AI-oriented FHE acceleration is evolving from independent primitive optimization to application, system and architecture-level co-design. CHAM [102] validates FPGA acceleration for encrypted linear algebra; FHE-CGRA [103] proves the advantages of reconfigurable dataflow for FHE-CNN inference. Cheetah [104] realizes full-stack optimization for private DNN inference via algorithm–hardware co-design; and HERACLES [105] verifies the potential of high-bandwidth, large-scale SIMD SoCs. As encrypted AI applications evolve from CNNs and logistic regression to Transformers and LLMs, future FHE accelerators need to optimize long-sequence attention, nonlinear approximation, KV cache management, batched inference and multi-chip collaborative execution.

4. Core Component Acceleration

Currently, mainstream FHE schemes are constructed over residue class polynomial rings $R_Q = \mathbb{Z}_q[x]/(x^N + 1)$. Specifically, the polynomial dimension is typically set to a power of two, while the modulus is generally chosen as a prime number and can alternatively be configured as a power of two [106]. For FHE schemes, polynomial multiplication and integer modular multiplication are the most time-consuming arithmetic units; thus, the hardware implementation of these two operations represents a crucial research focus in the development of FHE accelerators. Depending on whether the modulus is selected as a prime number or a power of two, research on polynomial multipliers is further categorized into two directions: polynomial multiplier designs based on NTT and INTT, and those based on the Toom–Cook (TC) or Karatsuba algorithms. The former exhibits a computational complexity of $O(N \log_2 N)$ and requires the modulus to be a prime number. In contrast, the latter has a computational complexity of $O(N^{\log_k(2k-1)})$ [107], where k denotes the number of branches decomposed by the Toom–Cook algorithm (referred to TC-k) and imposes no restrictions on the modulus. On the other hand, the integer modular multiplier serves as a core arithmetic component not only for polynomial multipliers but also for FHE accelerators.

Since different multipliers are compatible with distinct moduli and polynomial lengths, we construct a parameter selection decision diagram based on FHE parameters to standardize the presentation of such compatibility relationships. This diagram helps readers translate the conclusions of this survey into practical hardware design, as illustrated in Figure 3. The research status of these three aspects is elaborated separately in the subsequent sections.

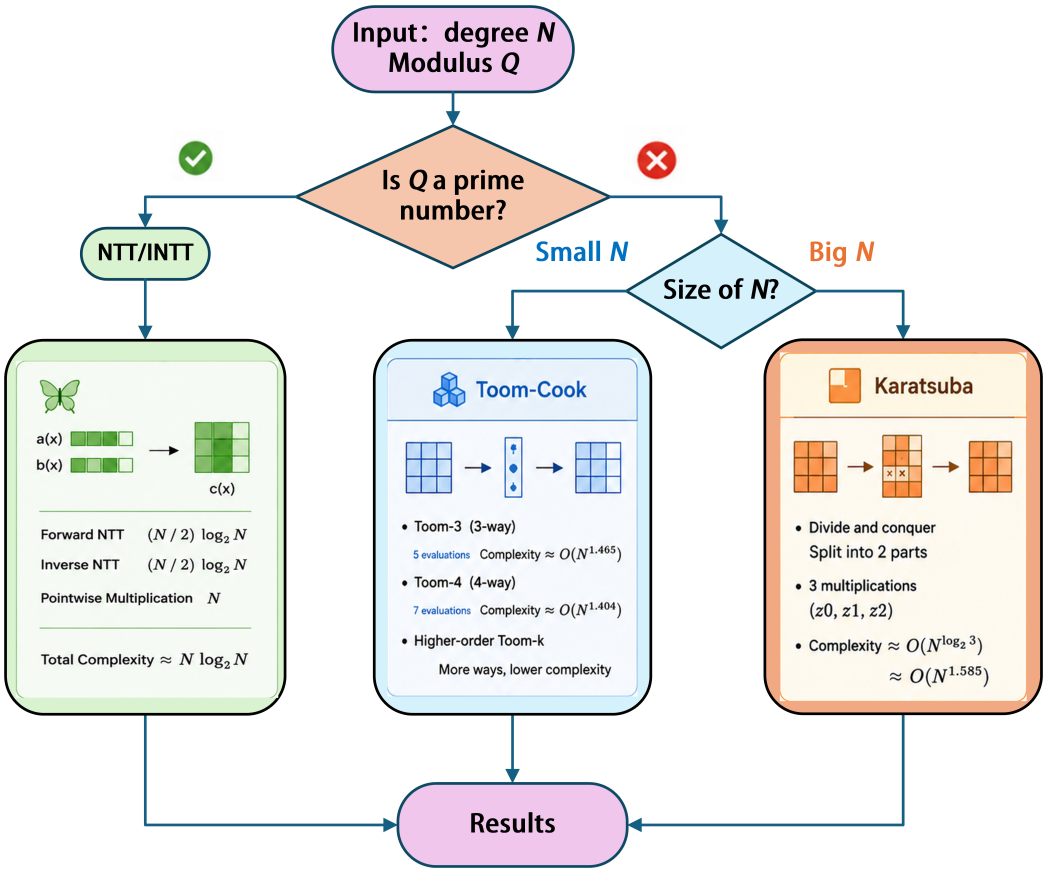


Figure 3. Decision Diagram for FHE Parameter Selection.

4.1. NTT/INTT-Based Polynomial Multipliers

NTT/INTT-based polynomial multipliers are the core arithmetic unit in most RLWE-based and NTRU-based FHE schemes. The main types and characteristics of NTT/INTT based polynomial multipliers are summarized as shown in Table 3.

Table 3. NTT/INTT-based polynomial multipliers.

Author	Algorithm	Hybrid	Platform	Strength	Weakness
Pöppelmann [108]	GS NTT	NO	FPGA	Supports dynamic parameter adjustment.	Inflexible, high resource consumption.
Xu [109]	CG NTT	YES	FPGA	High configurability, reduced critical path latency.	Fixed radix limitations, complex control.
Roy [110]	CT NTT	YES	FPGA	Two-level optimization balancing area and performance.	High parameter dependency, limited parallelism.
Du [111]	CT NTT	YES	FPGA	Improves operational efficiency and saves storage.	Limited butterfly parallelism, high storage overhead.
Pöppelmann [112]	CT NTT	YES	MCU	Low storage overhead with timing side-channel protection.	Poor parameter scalability.
Zhang [113]	CG NTT	YES	FPGA	Eliminates post-processing overhead, efficient.	Large twiddle factor storage.
Zeng [114]	CG NTT	YES	ASIC	High configurability, low resource utilization.	Complex control in high parallelism, limited security.
Chen [115]	CG NTT	YES	FPGA	Reduces control complexity, enables pipelining.	High resource consumption, limited scalability.
Banerjee [116]	CG NTT	YES	ASIC	Resists side-channel attacks, fully integrated.	Lower performance than dedicated designs, high utilization.
Longa [117]	CG NTT	YES	MCU	Flexible mixed radices.	Low throughput, security risks.
Phap [118]	CG NTT	YES	FPGA	High area efficiency with conflict-free memory.	Limited prime modulus compatibility, low throughput.
Mandal [119]	Winograd NTT	NO	FPGA	Reduced modular multipliers, low-latency.	More additions, high control complexity.
Ramyavani [120]	CG NTT	NO	FPGA	Low hardware utilization, high configurability.	Only optimized for small polynomials.
Bardia [121]	CG NTT	YES	FPGA	Extremely low latency, optimized resource use.	Large area overhead, limited prime modulus compatibility.
Zhao [122]	CG NTT	YES	RISC-V	Reduced data transmission latency, flexible.	Increased critical path latency, limited frequency.

Reducing computational complexity and twiddle-factor memory overhead remains a key issue in NTT/INTT-based polynomial multiplication. To avoid the doubled transform length caused by zero-padding, most designs employ Negative Wrapped Convolution

(NWC) for multiplication over polynomial rings. NWC enables the transform length to remain N instead of $2N$, but it introduces extra scaling-factor multiplications before NTT and after INTT, thereby increasing both arithmetic and storage overhead. For instance, Pöppelmann et al. [108] implemented a DIT NTT multiplier based on the Cooley–Tukey algorithm [109]. Although their NWC-based design reduced the transform length, one polynomial multiplication still required about $3(n \log n/2) + 5.5n$ cycles, where the additional $5.5n$ cycles mainly came from coefficient loading, point-wise multiplication, and post-scaling operations. For $n = 512$, the design consumed 10,014 cycles and 4 BRAMs, corresponding to 51.09 µs in a later normalized comparison. In 2014, Roy et al. [110] reduced the fixed cost of twiddle-factor generation by reorganizing the NTT loop structure and merged the NWC pre-processing into the forward CT NTT by modifying the initial twiddle factors. This optimization eliminated the pre-processing of NTT but could not be directly extended to INTT. Nevertheless, their compact Ring-LWE processor achieved encryption in 20 µs for $n = 256$ and 48 µs for $n = 512$, using only 1 DSP and 2–3 BRAMs. This idea was later inherited by several NTT-based polynomial multipliers [111,113]. Du et al. [111] further reduced the polynomial-multiplication cost from $8N + 1.5N \log N$ to $2N + 1.5N \log N$, and compressed the stored constant factors from $4N$ to $2.5N$, saving 37.5% ROM. For $n = 512$, their design achieved 8806 cycles and 26,913 multiplications on Spartan-6, improving throughput by 1.38×. However, because the constants $N^{-1}\psi^{-i}$ still had to be merged into CT INTT, the post-processing stage was not fully removed. Pöppelmann et al. [112] addressed this limitation from another direction by adopting a GS-based DIF-INTT, in which the ψ^{-i} factors were folded into the INTT twiddle factors; on an 8-bit ATxmega128, their optimized implementation completed RLWE encryption/decryption in 27/6.7 ms for $n = 256$, but the N^{-1} scaling remained an inherent cost unless absorbed into precomputed constants. Zhang et al. [113] finally merged both γ^{-i} and N^{-1} into the GS-INTT butterfly, eliminating the INTT post-processing overhead and reducing modular multiplications by 18.2% and 30.8% for 512-point NTT/INTT, but still required $2N$-scale twiddle-factor storage, showing that post-processing elimination and memory overhead remain a coupled optimization problem.

Most of the aforementioned designs [108,111–113,123] and many other studies [114,124–126] mainly adopted in-place CT NTT or GS INTT structures. Although such in-place architectures can write data of each NTT/INTT stage back to the original addresses and save memory overhead, the memory access pattern becomes highly complex and varies with the number of accelerator cores in a multi-core architecture, which is detrimental to the scalability of multi-core systems. To simplify memory access patterns, in 2014 and 2019, Chen et al. [115] and Banerjee et al. [116] respectively proposed Constant-Geometry (CG) algorithm-based NTT architectures to accelerate polynomial multiplication. By adopting an out-of-place computation scheme, each stage of CG NTT and INTT maintains a consistent structure regardless of the number of accelerator cores, thus, facilitating scalability in multi-core architectures. However, compared with in-place NTT, the data memory overhead of CG NTT is doubled. How to improve area efficiency and avoid bit-reversal operations is another important research topic for NTT/INTT-based polynomial multipliers. Existing designs [111,127–129] mostly adopt non-configurable NTT structures or partially configurable NTT structures to save hardware resources and improve area efficiency. However, since DIT NTT typically uses bit-reversed input and natural output, while DIF INTT uses natural input and bit-reversed output, separate bit-reversal operations must be performed on input and output data. Longa et al. [117] addressed this problem algorithmically by combining CT-based forward NTT with GS-based inverse NTT, allowing the forward transform to output bit-reversed data and the inverse transform to consume it directly. With an optimized modular reduction, their AVX2 version implementation

achieved 9.1 k/9.7 k cycles, giving 1.21–1.90× speedups over prior constant-time implementations. However, directly mapping CT and GS to hardware may require separate butterfly datapaths and additional area overhead. To improve area efficiency, from 2019 to 2021, the research group led by Prof. Chen et al. [115] (from the group of Prof. Zeng at Fudan University), Banerjee et al. [116], and Mert et al. [130] proposed reconfigurable butterfly designs supporting simultaneous NTT, INTT, and point-wise multiplication, which reduce hardware overhead while avoiding bit-reversal operations. Phap et al. [118] proposed an area-efficient NTT architecture for FHE schemes, centered on a high-throughput butterfly array that communicates with a single data memory module through conflict-free memory access. A twiddle factor generator was also developed to reduce storage by 99.43% and 12×/13× speedups were achieved over SEAL NTT/INTT for 32 RNS primes. Xu et al. [109] reused the NTT module for preprocessing, NTT/INTT, point-wise multiplication, and post-processing. Their $N = 512$ multiplier achieved 13.32 µs latency with 485 slices, 5 BRAMs, and 18 DSPs, showing the effectiveness of module reuse for resource-sensitive designs. More recently, Mandal et al. [119] explored Winograd-based higher-radix NTT as an alternative to CT/GS structures. Their radix-16 Winograd design reduced multiplier demand and achieved 58-cycle NTT/INTT latency for Dilithium, demonstrating a promising direction for low-latency higher-radix NTT implementation. Ramyavani et al. [120] compared several Barrett modular-reduction variants in terms of resource utilization and accuracy, and validated a three-stage radix-2 NTT multiplier on FPGA, completing an $n = 8$ polynomial multiplication in only 9 clock cycles. Although this design demonstrates the benefit of reduction-level optimization, its very small transform size makes it more suitable as a proof-of-concept than as a direct solution for practical PQC parameters. In contrast, Bardia et al. [121] proposed ParallelNTT, a fully parallel FPGA architecture for both forward NTT and INTT in ML-KEM and ML-DSA. By removing RAM-based sequential coefficient access and feeding all coefficients to each NTT stage simultaneously, ParallelNTT reduces transform latency from $O(n)$ or $O(n \log n)$ hardware schedules to $O(\log n)$, and achieves 24× and 21× latency improvement over prior FPGA designs. However, this ultra-low latency is obtained at the cost of high parallel hardware usage, e.g., 128 DSPs for ML-KEM and 256 DSPs for ML-DSA. In 2025, Zhao et al. [122] proposed an innovative hardware architecture that tightly couples the NTT unit with the load-store unit (LSU) in a RISC-V processor pipeline and employs a mixed-width datapath method to effectively reduce data transfer time. Compared with prior designs, the architecture minimizes data transfer latency while enhancing computational flexibility and scalability.

Discussion: Although existing NTT/INTT-based polynomial multipliers have reduced part of the pre-processing, post-processing, and bit-reversal overhead, several design trade-offs remain. In-place CT NTT or GS INTT architectures are memory-efficient, but their stage-dependent access patterns become more complex as the number of butterfly units or processing cores increases, which may limit scalability in parallel architectures. Meanwhile, NTT and INTT usually require different twiddle-factor orders or addressing schemes. Although memory sharing is possible, it often requires additional scheduling and control logic. Combining DIT NTT with DIF INTT can avoid explicit bit-reversal operations, but their different butterfly dataflows may increase hardware complexity. Reconfigurable butterfly units improve datapath reuse and area efficiency by supporting NTT, INTT, and point-wise multiplication within a shared structure, yet they may still introduce extra control cycles or fail to remove all preprocessing-related overhead, depending on the algorithm and memory organization. Therefore, efficient NTT/INTT polynomial multiplier design still requires a careful balance among arithmetic complexity, twiddle-factor storage, memory-access regularity, and butterfly-unit reuse, especially for FHE accelerators with large polynomial degrees and multiple RNS moduli.

4.2. TC/Karatsuba-Based Polynomial Multipliers

Although NTT/INTT-based polynomial multipliers exhibit lower computational complexity, they impose strict practical requirements that the polynomial dimension N must be a power of two and the modulus must be a prime number satisfying $q \equiv 1 \pmod{2N}$. Consequently, additional zero-padding operations are required for non-power-of-two polynomials, and the modular multiplication unit becomes more intricate, leading to increased hardware resource overhead. In contrast, TC/Karatsuba-based polynomial multipliers have no restrictions on polynomial dimension or modulus selection, enabling better adaptability to diverse cryptographic parameters. Furthermore, the modulus can often be chosen as a power of two [30], which simplifies complex and time-consuming operations such as modular multiplication and division into simple shift operations. Therefore, TC/Karatsuba-based polynomial multipliers have attracted increasing research interest. The main types and characteristics of TC/Karatsuba based polynomial multipliers are summarized as shown in Table 4.

Table 4. Karatsuba/TC-based polynomial multipliers.

Author	Algorithm	Platform	Strength	Weakness
Migliore [131]	Karatsuba-3	FPGA	Optimized for low latency.	Limited flexibility, lacks dynamic reconfiguration, reduced throughput.
Zhu [132]	8-level Iterative Karatsuba	FPGA+ASIC	Balanced complexity and efficiency, high resource utilization.	High recursive decomposition overhead, lacks side-channel protection.
Wong [133]	1-level Karatsuba	FPGA	High hardware efficiency in negative cyclic convolution, high DSP utilization.	Limited parallelism, lacks side-channel security.
Wang [134]	TCPM (TC-2/3/4)	FPGA	Highly configurable, high parallel efficiency and computing speed.	Complex control logic.
Karmakar [135]	TC + Karatsuba + Schoolbook	Cortex-M4	Extremely low memory footprint, high algorithm adaptability.	High complexity, lacks side-channel protection.
Mera [136]	3-level Karatsuba	AVX	Optimized memory usage, resistant to timing attacks.	Complex control logic, limited scalability.
Kannwischer [137]	TC-4 + Multi-level Karatsuba	Cortex-M4	Improved performance, efficient and stable memory usage.	Limited scalability, lacks side-channel protection.
Paksoy [138]	TC-4	Cortex-M4	High versatility, improved performance.	Poor scalability, lacks side-channel protection.
Mera [139]	TC-4	FPGA	High throughput, high access efficiency.	Limited scalability, lacks side-channel protection.
Ghosh [140]	Striding TC-4	ASIC	Low memory footprint, high algorithm adaptability.	Limited parallel performance, low algorithm flexibility.
Li's group [141,142]	TC-3	ASIC	Modular design, high scalability.	High hardware resource usage, lacks side-channel protection.
Ma [143]	TC-3	FPGA	Improved operational efficiency, reduced latency.	Limited operating frequency, insufficient side-channel protection.
Vasanthi [144]	Hybrid Karatsuba	FPGA + ASIC	Area optimization, reduced latency, high versatility.	High complexity, insufficient side-channel protection.

Optimizing the pre-processing and post-processing stages is one of the key research focuses for Karatsuba-based polynomial multipliers. Migliore et al. [131] investigated Karatsuba-based polynomial multiplication for SHE/RLWE schemes under a realistic ARM-FPGA co-design scenario. By exploiting the lower data dependency of Karatsuba and the symmetry of its pre-/post-computation stages, they optimized coefficient scheduling and reduced recombination overhead. For $n = 1024$, the optimized schedule reduced the operation count from 3583 to 3071, clearly lower than FFT with 4158 operations. Under limited embedded multipliers, Karatsuba achieved up to 58% speedup. They later extended this idea to a 9-level recursive Karatsuba-based FV accelerator [128], indicating that Karatsuba can outperform FFT/NWC in resource-constrained HE accelerators despite its weaker asymptotic complexity. Zhu et al. [132] further optimized this direction for Saber with an 8-level hierarchical Karatsuba structure, reducing degree-256 polynomial multiplication from 65,536 to 6521 coefficient multiplications and completing one multiplication in 81 cycles. Through register and adder reuse in pre-processing and register elimination in post-processing, their design achieved about 3.16× speedup for degree-256 multiplication. Nevertheless, all the aforementioned designs cannot avoid polynomial modular reduction operations. Wong et al. [133] proposed a hybrid Karatsuba architecture

for RLWE-oriented polynomial multiplication, integrating modular reduction into the recombination stage to avoid a separate reduction unit. This design achieved more than 2.09× speedup over the state-of-the-art architecture. However, this optimization is limited to Karatsuba-based polynomial multipliers and cannot be extended to Toom–Cook-based designs. To improve Toom–Cook efficiency, Wang et al. [134] proposed TCPM, a reconfigurable Toom–Cook multiplier supporting Toom–Cook-2, 3, and 4 on a unified architecture. By introducing a compressed and fused post-processing algorithm, TCPM reduces post-processing arithmetic operations by at least 33.33%. When implemented on a Xilinx VC709 FPGA, it achieves 2.47×–50.11× speedup over prior designs. Compared with Wong et al.'s Karatsuba-specific reduction fusion, TCPM offers better configurability and algorithm-level flexibility, while its efficiency mainly comes from reducing interpolation/post-processing complexity rather than eliminating modular reduction itself.

The Karatsuba algorithm is a special case of the Toom–Cook (TC) algorithm when decomposed into two branches [145]. Compared with algorithms such as TC-3/4, it exhibits higher computational complexity and finer decomposition granularity. This implies that for the same polynomial dimension, more iterations are required and the memory access pattern becomes more complicated. In contrast, TC-3/4 algorithms feature lower computational complexity, coarser decomposition granularity, and fewer iterations, and have, therefore, attracted considerable attention from cryptographic hardware researchers.

Algorithm and memory optimization for TC-based polynomial multipliers on embedded accelerators represents an important research direction. Karmakar et al. [135] presented a TC-based polynomial multiplier for Saber on ARM Cortex-M platforms, decomposing 256-dimensional polynomial multiplication into one TC-4 layer, two Karatsuba layers, and a final Schoolbook layer. Building on Roche's space-efficient Karatsuba method [146], which preserves Karatsuba's subquadratic complexity while reducing auxiliary space to $O(\log n)$, they combined vertical scanning in TC-4 evaluation/interpolation, in-place Karatsuba, DSP-assisted half-word cross multiplication, and non-adjacent coefficient packing. This design achieved 65,459–66,692 cycles for one 256 × 256 multiplication on Cortex-M4, outperforming several reported NTT-based implementations. Compared with this bottom-up optimization strategy, Mera et al. [136] further reduced the upper-level TC overhead by treating evaluation and interpolation as linear transformations and introducing lazy interpolation plus precomputation for Saber [147]. Their method improves matrix-vector polynomial multiplication by 12–37% across platforms, with AVX2 matrix-vector multiplication gaining 29.3–36.6%, while also reducing Cortex-M4 memory usage by 2.6–5.7 KB. Overall, [135] mainly optimizes the instruction-level and memory-access efficiency of the TC/Karatsuba/Schoolbook hierarchy, whereas [136] amortizes TC pre- and post-processing at the module-lattice matrix-operation level, giving a clearer time–memory trade-off. In addition, Kannwischer et al. [137] and Paksoy et al. [138] respectively proposed optimized implementations of TC polynomial multipliers based on the Cortex-M4 accelerator. Although these designs adopt a multi-level hybrid structure of TC, Karatsuba, and Schoolbook multiplication, their optimization approaches are mainly suitable for embedded accelerators. Limited by the sequential execution of embedded accelerators, the parallel processing performance of TC polynomial multipliers cannot be further improved.

Optimizing interpolation operations and enhancing memory access efficiency on FPGA or ASIC platforms is another important research direction. Building on their prior time–memory trade-off study [136], Mera et al. [139] presented the first FPGA-based Toom–Cook polynomial multiplier for Saber. The design applies one TC-4 decomposition to a 256-coefficient multiplication and executes the seven 64-coefficient products in parallel using compact schoolbook multipliers. With vertical coefficient scanning, lazy interpolation, and a coefficient-parallel memory layout, the co-processor achieves a 26.7× speedup for

polynomial multiplication and 5.4×/5.4×/7.0× speedups for Saber key generation, encapsulation, and decapsulation, respectively. However, its interpolation stage still contains serial data dependencies and irregular write patterns, which limit fine-grained parallel scalability. Ghosh et al. [140] further extended this TC-based Saber acceleration to silicon by adopting a strided Toom–Cook data layout, which directly supports negacyclic reduction and reduces on-chip memory demand. Their 65-nm Saber ASIC occupies 0.158 mm^2, uses about 10 KB SRAM. Compared with prior PQC ASICs, it reports 1.75× lower area, about 4× lower memory, and 1.58× higher multiplication energy efficiency. Nevertheless, its TC datapath remains mainly optimized for fixed Saber parameters, and its algorithmic flexibility is, therefore, limited. Furthermore, the research group led by Prof. Li at Tsinghua University [141,142] designed division-free TC-based large-integer modular multipliers for integer multiplication, but these are mainly applicable when the modulus is a prime number. Ma et al. [143] optimized the FPGA implementation of Streamlined NTRU Prime by applying a TC-3 polynomial multiplier to the encapsulation and decapsulation stages. This method reduces large-integer multiplications from nine to five and lowers the multiplication complexity from $O(n^2)$ to $O(n^{1.465})$. On the Intel Cyclone IV FPGA platform, the design improves encapsulation and decapsulation speed by 26%, while the Toom–Cook-3 module occupies only 205 LUTs, 128 FFs and 1 DSP, indicating that it provides effective acceleration with limited hardware overhead. Vasanthi et al. [144] proposed HRM, a heterogeneous recursive multiplier mixing KM, OKM and SBM at different recursive layers. It adopts low-latency OKM for high-order segments and area-efficient SBM for low-order suboperations. Benchmarked on FPGA and 45 nm ASIC, HRM reduces delay by up to 29.53% and improves ADP by 45.66% versus mainstream M-term Karatsuba variants, with 13.49% less LUT consumption on Zynq platforms.

Discussion: TC/Karatsuba-based polynomial multipliers have emerged as an alternative to FFT/NTT-based designs for RLWE-oriented FHE, offering strong algorithmic flexibility through multi-level recursion combining Karatsuba, Toom–Cook, and Schoolbook multiplication. However, their performance is fundamentally constrained by execution and dataflow characteristics: on embedded processors, the sequential execution model limits exploitation of inherent recursion parallelism, while FPGA/ASIC implementations are typically restricted to single-level TC-4 or shallow hybrid structures, which improves hardware simplicity but reduces scalability to higher polynomial dimensions. In addition, strong data dependencies in interpolation and recombination stages introduce irregular memory access and serial bottlenecks, further restricting multi-core parallel efficiency. Despite these limitations, TC/Karatsuba multipliers remain attractive for lightweight FHE accelerators due to their adaptability to resource constraints, lower hardware overhead than FFT/NTT-based approaches at moderate dimensions, and favorable area–latency–memory trade-offs.

4.3. Integer Modular Multipliers

The integer modular multiplier is a key arithmetic component in polynomial multiplication and FHE accelerators. Its performance and flexibility in modulus selection directly determine the performance of FHE accelerators and the choice of cryptographic parameters. An integer modular multiplier mainly consists of two parts, which contains an integer multiplier and a modular reduction unit. At present, mainstream integer modular multipliers mainly include the Barrett multiplier, Shoup multiplier, Montgomery multiplier, Solinas prime multiplier, and multipliers for specific prime moduli. Öztürk et al. [85] proposed a 32-bit Barrett modular multiplier to accelerate the LTV scheme. By precomputing the final conditional subtraction and using a multiplexer for selection, the output latency of conditional subtraction was effectively reduced. This optimized multiplier structure was adopted and further extended to support reconfigurable operations and achieved a clock

frequency of 250 MHz on the Xilinx Virtex-7 FPGA. Compared to the Montgomery multiplier, this structure reduces the computational latency by approximately 30% and improves the throughput of the homomorphic multiplication unit by 1.45× to 5.51×, demonstrating superior performance in high-speed homomorphic encryption scenarios. Cousins et al. [61] proposed an interleaved modular multiplier based on Barrett reduction, which represents long input words as multiple short words and processes them in a pipelined, stage-by-stage manner to shorten the critical path of wide-word modular multiplication. This design enables the multiplier to operate at a high clock frequency of 350 MHz. Compared with the traditional direct multiplication method, this structure significantly improves the computational efficiency of modular multiplication. Roy et al. [62] proposed a DSP-efficient 31-bit Barrett modular reduction unit for the BFV scheme, utilizing six DSPs and partial LUTs to balance area and performance. This design enables the polynomial arithmetic unit to operate at 100 MHz. The unit's integration into a parallel NTT architecture allows the processor to complete a homomorphic multiplication in 3.36 s excluding memory access overhead. Hao et al. [148] proposed a Barrett modular multiplication algorithm based on the Toom–Cook method, which demonstrated superior area-efficiency in ASIC implementations. Compared to existing works, the design achieves significant performance improvements under 65 nm and 90 nm processes with Area-Time-Product (ATP) reductions of 8.7% and 17.5% respectively. The algorithm eliminates redundant factors through optimized precomputation, enabling efficient hardware acceleration for large-integer multiplications in ECC and RSA protocols. Although the Barrett modular multiplier supports modular multiplication between two arbitrary variables and achieves high modular reduction efficiency, partial values can be precomputed to reduce the number of multiplications if one input is a constant. To improve the efficiency of modular multiplication between a variable and a constant, the Shoup modular multiplier was introduced separately in the NTL library [149]. Riazi et al. [65] proposed dedicated Barrett and Shoup modular multipliers for variable-variable and twiddle factor multiplications in NTT/INTT, respectively. However, instantiating these two distinct units increased hardware resource overhead. This design choice leads to significant resource consumption, with the NTT module alone utilizing 30% of DSP units and 42% of ALMs on an Arria 10 FPGA, and the separate multipliers create timing bottlenecks, limiting the maximum clock frequency to 275 MHz. Kim et al. [150] proposed fully pipelined DSP-based Barrett and Shoup modular multipliers, optimizing the arithmetic structure of integer multiplication and forwarding partial values between pipeline stages to effectively reduce hardware overhead and latency. Their design achieved a 14× higher throughput at the same frequency and 3× higher throughput/DSP compared to previous non-fully pipelined Barrett designs, with a specific 62-bit Barrett implementation reaching 425 MHz using only 12 DSP slices. In 2024, Mazonka et al. [151] extended the Shoup multiplier by incorporating the design principle of Barrett reduction, creating a "Generalized Shoup" architecture that supports arbitrary operands. While this generalization increased the area by approximately 30% compared to the standard Barrett multiplier, it allowed a single unit to handle both specialized (Shoup) and general (Barrett) operations, resulting in a critical path delay ∼20% faster than the traditional Barrett implementation. Although the Shoup multiplier offers higher processing speed, it requires one input to be a constant, which limits its application scope.

Mert et al. [63] proposed a word-level Montgomery modular multiplier optimized for NTT operations, exploiting the structure of NTT-friendly primes ($q = q_H \cdot 2^w + 1$) to simplify the Montgomery constant to $\mu = -1$, thereby eliminating a dedicated multiplication in the reduction step. Implemented on a Xilinx Virtex-7 FPGA at 200 MHz, the iterative architecture completes a polynomial multiplication in 0.96 µs and encryption/decryption in 1.4 µs and 1.24 µs, respectively. Integrated into a hardware/software co-design frame-

work with PCIe communication, the design achieves 12× and 7× latency speedups over pure software SEAL implementations for encryption and decryption, while demonstrating the lowest NTT/polynomial-multiplication latency among comparable FPGA designs. Hongshuo et al. [152] improved the classical Montgomery modular multiplication algorithm by proposing an interleaved pipeline architecture without PE reuse, which shortens intermediate-data paths and enhances area efficiency. Implemented on Xilinx Virtex-7 FPGA, the design completes 1024-bit and 2048-bit modular multiplication in 1.03 µs and 2.13 µs, respectively, with ATP values of 3.605 and 14.454. Compared with existing FPGA designs, it maintains competitive latency while achieving better overall area–time efficiency, making it suitable for low-cost, high-performance IoT cryptographic applications. Roy et al. [153] proposed an improved Montgomery modular multiplier leveraging hybrid recursive Karatsuba multiplication (HRKM) and optimized schoolbook multipliers (OSBM) to achieve single-cycle execution. This design significantly reduces resource utilization and latency on both FPGAs and ASICs. Implemented on a Virtex-6 FPGA, it achieves a 92.766% reduction in Area-Time Product (ATP) compared to state-of-the-art designs. On a Virtex UltraScale+ XCVU9P FPGA, the ATP reduction reaches 95.130%. Furthermore, ASIC synthesis using UMC 65-nm technology demonstrates a 22.173% improvement in Power-Time Product (PTP) over existing works, highlighting its superior energy efficiency for resource-constrained cryptographic systems. Wang et al. [154] proposed a parallel and pipelined Montgomery multiplier (PPMM) for IoT devices, which optimizes algorithm scheduling to reduce data dependencies and overall clock cycle count. Implemented on a Xilinx Virtex-7 FPGA over general GF(p) for field sizes 256, 384, and 512 bits, the design completes a high-radix modular multiplication in only 0.123 µs, 0.150 µs, and 0.190 µs, respectively. Compared to other FPGA implementations, the proposed design demonstrates superior performance in terms of speed and throughput. Specifically, the 384-bit PPMM achieves a throughput nearly twice that of other designs, making it well-suited for security-intensive applications. Cao et al. [155] adopted a Solinas prime modular multiplier with modulus $2^{64} - 2^{32} + 1$ to accelerate modular multiplication in the Fast Fourier Transform (FFT). In the context of their low-latency integer-FFT multiplier architecture for FHE, this specific modulus was chosen because it enables highly efficient modular reduction. The design achieved a clock frequency of 161.276 MHz on a Xilinx Virtex-7 FPGA and demonstrated superior performance in high-speed homomorphic encryption scenarios. Zijlstra et al. [156] proposed an efficient Solinas modular multiplier for moduli $q = 2^{h_1} - 2^{h_2} + 1$, leveraging the congruence $2^{h_1} \equiv 2^{h_2} - 1 \pmod{q}$ to replace complex divisions with simple bit shifts, additions, and subtractions. This optimization is integrated into a high-speed NTT architecture that achieves a clock frequency of 250 MHz on a Xilinx Artix-7 FPGA. Specifically, it utilizes only 1 DSP slice and 1624 slices, demonstrating a favorable balance between speed and hardware resource utilization for RLWE-based cryptography. Zhang et al. [113] proposed an optimized modular multiplication implementation for the specific prime modulus 12,289, leveraging the property $2^{14} \equiv 2^{12} - 1 \pmod{12{,}289}$ to minimize hardware resource overhead. Their design achieves a four-stage pipeline with a critical path delay optimized for high-speed operation, consuming only 2 DSP slices and 6781 LUTs on a Xilinx Artix-7 FPGA. This results in an Area-Time Product (ATP) that is approximately 3× smaller than state-of-the-art designs, demonstrating superior efficiency for the NewHope-NIST key encapsulation mechanism. However, both Solinas multipliers and special-purpose prime multipliers impose strict constraints on the modulus form. Moreover, when the modulus is small, the number of available Solinas primes or special primes is limited, which is unfavorable for implementing FHE schemes with diverse parameter sets. The main types and characteristics of Integer Modular Multipliers are summarized as shown in Table 5.

Table 5. Typical modular multipliers.

Multiplier Type	Literature	Scheme	Platform	Characteristic Analysis
Barrett modular multiplier	Öztürk [85]	LTV	FPGA	High versatility and efficiency in modular reduction with large optimization potential; however, long critical path for large bit-width leads to high resource consumption.
	Roy [62]	BFV	FPGA	Resource-efficient design with a good balance between area and speed.
	Hao [148]	Toom–Cook	ASIC	Excellent area efficiency.
Shoup modular multiplier	Riazi [65]	NTT	CPU + FPGA	Optimized for different scenarios, but dual instantiation leads to high hardware overhead.
	Kim [150]	CKKS	FPGA	Low resource overhead and latency; however, requires one operand to be constant, limiting applicability.
Montgomery modular multiplier	Mert [63]	BFV	CPU + FPGA	Fewer modular reductions, but additional overhead for domain conversion.
	Li [152]	IP-MMM	FPGA	Short data path, suitable for high-performance and low-cost IoT.
	Roy [153]	HRKM + OSBM	FPGA + ASIC	Single-cycle operation with low resource usage and latency.
	Wang [154]	PPMM	FPGA	Low cycle count and high operating frequency.
Solinas prime modular multiplier	Cao [155]	FFT/IFFT	FPGA	Simplifies reduction using special modulus forms, but requires specific modulus structure.
	Zijlstra [156]	NTT/INTT	FPGA	Simple hardware (shift/add/sub), but requires special modulus forms.

Discussion: Based on the above analysis, different integer modular multipliers exhibit distinct trade-offs in FHE accelerator design. Solinas-based and special-prime multipliers enable highly efficient reductions by exploiting specific modulus structures, but their strict constraints on admissible primes significantly limit parameter flexibility in RLWE-based schemes. In contrast, Barrett and Montgomery multipliers support general moduli and variable–variable multiplication, making them more widely applicable; however, Montgomery designs introduce additional overhead due to domain conversions, while Barrett implementations are often less optimized for specific operand structures. The Shoup multiplier improves efficiency for variable–constant multiplication, yet its applicability is restricted by the requirement that one operand must be a fixed constant, reducing its generality. Overall, these approaches reflect a fundamental trade-off between computational efficiency and modulus/operand flexibility, where highly optimized schemes sacrifice universality, whereas general-purpose multipliers incur additional computational or architectural overhead.

5. Challenges and Future Outlook

By summarizing the architectures of existing FHE accelerators, the following conclusions can be drawn:

- Existing FHE accelerators adopt dedicated accelerator cores or hardware components for different FHE primitives, such as NTT/INTT, modular arithmetic, modulus raising/reduction, and rescaling. Although these dedicated accelerators effectively improve the computational performance of individual FHE primitives, they also restrict the sharing and reuse of hardware resources among different acceleration modules. A large number of common processing units still exist inside these independent accelerators, and most of them do not need to operate simultaneously. Consequently, the discrete accelerator architecture reduces hardware resource reusability and utilization efficiency.
- Most existing FHE accelerators are tailored for a specific FHE scheme with extensive customized design strategies. Although they can be extended to support one or two additional FHE schemes of the same category, the fixed hardware unit structure

results in insufficient flexibility and limits the performance optimization of compatible algorithms. Meanwhile, current FHE accelerators are mainly designed for finite-depth computational circuits, lack native support for bootstrapping algorithms, and cannot accommodate unbounded-depth computational circuits.

- Most existing FHE accelerator designs are optimized for general-purpose application scenarios. For specialized domains such as neural network training, multi-party secure computation, and federated learning, current accelerators fail to fully satisfy the requirements on homomorphic computation depth, cryptographic algorithm parameters, hardware area, power consumption, and other metrics. Therefore, the hardware optimization effectiveness and functional matching degree remain to be further improved.

Further research can be explored in the following directions in the future:

- Design of reconfigurable hardware accelerators with shared arithmetic and configurable dataflow. Most existing FHE schemes differ in ciphertext formats, modulus management, and evaluation procedures, but they still share a substantial set of low-level arithmetic kernels, including polynomial multiplication, modular addition/subtraction, integer modular multiplication, NTT/INTT, point-wise multiplication, permutation, and memory-intensive key-switching-related data movement. Therefore, future FHE accelerators should evolve from isolated operator-specific or scheme-specific cores toward a layered reconfigurable architecture. At the bottom layer, modular arithmetic units, butterfly arrays, NTT/INTT engines, point-wise multipliers, RNS channels, and on-chip memory banks can be shared across BFV, BGV, CKKS, and partly across FHEW/TFHE-like schemes. At the middle layer, configurable control logic, microcoded instruction scheduling, programmable modulus-reduction modes, flexible radix selection, and parameterized polynomial dimensions should be supported to accommodate different values of N, q, RNS bases, decomposition bases, and security levels. At the top layer, scheme-specific dataflows such as BFV base extension and scaling, CKKS rescaling and approximate modulus conversion, and FHEW/TFHE accumulator updates should be retained as specialized microprograms or dedicated functional extensions. In this way, common arithmetic resources can be reused without sacrificing the optimization opportunities of individual schemes. Existing programmable and configurable designs such as F1, Medha, ARK, Poseidon, and scheme-switching accelerators have already demonstrated the value of programmability, memory reuse, runtime data generation, and cross-scheme support, but future designs still need finer-grained sharing mechanisms, lower control overhead, stronger parameter scalability, and better integration between the arithmetic datapath and the memory system [66,67,69–71,75].
- Design of bootstrapping-oriented accelerators for unbounded-depth computation. Bootstrapping remains the key operation that turns leveled FHE into FHE supporting arbitrary circuit depth, and it is also the dominant performance bottleneck in many practical deployment scenarios. For BGV, BFV, and CKKS schemes, bootstrapping introduces large parameter sets, intensive NTT/INTT computation, RNS base conversion, key switching, slot rotation, linear transformations, modulus raising/reduction, and rescaling. Therefore, bootstrapping accelerators for leveled schemes should prioritize high-throughput NTT/INTT pipelines, conflict-free multi-bank memory, efficient rotation/key-switching dataflows, large evaluation-key storage or on-the-fly key generation, and precision-aware arithmetic units for CKKS. For FHEW/TFHE-like bit-bootstrappable schemes, the architectural focus is different: the main bottlenecks lie in blind rotation, homomorphic accumulator updates, polynomial–vector and polynomial–matrix multiplication, FFT/NTT over torus or ring representations, and the bandwidth required by bootstrapping keys. Thus, these schemes require streaming

accumulator architectures, compact bootstrapping-key layouts, fixed-point or integer-friendly transform units, and low-latency gate-level evaluation pipelines. Recent accelerators such as CraterLake, MATCHA, FPT, and Poseidon indicate that bootstrapping support is becoming a central design goal, but future work should further clarify which modules can be shared with leveled homomorphic evaluation and which must remain specialized for bootstrapping. In particular, NTT/INTT, modular arithmetic, memory controllers, permutation networks, and key-switching support are suitable for sharing, whereas CKKS precision management, TFHE blind rotation, torus arithmetic, circuit bootstrapping, and scheme-switching procedures require dedicated architectural optimization [68,71,73–75].

- Research on application-oriented FHE accelerators based on hardware–software co-design. Although general-purpose FHE accelerators improve the performance of basic homomorphic primitives, practical applications impose different requirements on latency, throughput, multiplicative depth, precision, memory capacity, communication overhead, energy efficiency, and supported scheme types. Therefore, future FHE accelerators should be designed according to application workload characteristics rather than only according to cryptographic primitives. For privacy-preserving neural-network inference and encrypted machine learning, accelerator design should emphasize CKKS/BFV-friendly SIMD packing, ciphertext rotation, key switching, rescaling, NTT/INTT, matrix-vector multiplication, convolution, nonlinear approximation, and bootstrapping when the circuit depth exceeds the leveled parameter budget. Recent AI-oriented accelerators such as CHAM, FHE-CGRA, Cheetah, and HERACLES demonstrate the effectiveness of customized encrypted linear-algebra datapaths, reconfigurable dataflow mapping, full-stack private-inference optimization, and large-scale SIMD architectures [102–105]. For federated learning, the design focus should extend from accelerating individual encryption or aggregation operations to optimizing the complete secure-training pipeline, including ciphertext packing, secure aggregation, communication compression, homomorphic matrix–vector computation, and cross-institutional feature interaction. Representative works such as FedBit and SAFE show that cross-layer packing, compression-aware data organization, and matrix-vector-oriented FPGA dataflows can effectively improve privacy-preserving federated learning efficiency [99,101]. For edge and IoT scenarios, accelerators should further prioritize low area, low power, compact parameter sets, lightweight modular arithmetic, client-side encoding/encryption acceleration, and efficient edge–cloud task partitioning. Existing designs such as HHEML, Pasta on Edge, embedded SEAL/CKKS NTT accelerators, ReFHE-NTT, and compact CKKS client-side accelerators indicate that hybrid homomorphic encryption, resource-driven NTT units, and unified client-side FHE pipelines are promising directions for resource-constrained deployment [94–98]. Overall, application-specific FHE accelerators should jointly optimize cryptographic parameters, algorithmic transformations, data layout, memory hierarchy, communication patterns, and hardware datapaths, so as to promote FHE acceleration from isolated primitive-level speedup toward practical privacy-preserving systems.

6. Conclusions

Recently, hardware research on FHE has been developing rapidly. This paper focuses on the latest advances in this field and systematically summarizes the relevant achievements of FHE schemes in recent years. The research progress of the overall architecture of FHE hardware, polynomial multipliers, and integer modular multipliers is comprehensively reviewed and analyzed. Finally, potential deficiencies of fully homomorphic encryption

accelerators are summarized, and the future development directions are proposed, aiming to provide useful references for subsequent research.

Author Contributions: Conceptualization, Y.S. and K.Z.; Methodology, Y.S. and K.Z.; Supervision, W.Z.; Resources, W.Z.; Formal Analysis, J.W. and J.H.; Visualization, J.W. and J.H.; Investigation, J.W. and J.H.; Writing—Review & Editing, C.Y.; Project Administration, C.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article. All analyses and conclusions presented in this work are derived entirely from existing published literature and publicly available theoretical models, and no original experimental datasets or raw measurement data were generated during the research.

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

References

1. Li , Z.Y.; Gui, X.L.; Gu, Y.J.; Li, X.S.; Dai, H.J.; Zhang, X.J. Homomorphic encryption technology and its application in cloud computing privacy protection. *J. Softw.* **2018**, *29*, 1830–1851.
2. Park, J.; Tibouchi, M. SHECS-PIR: Somewhat homomorphic encryption-based compact and scalable private information retrieval. In *European Symposium on Research in Computer Security* ; Springer: Berlin/Heidelberg, Germany, 2020; pp. 423–442.
3. Wang, X.; Luo, T.; Li, J. An efficient fully homomorphic encryption scheme for private information retrieval in the cloud. *Int. J. Pattern Recognit. Artif. Intell.* **2020**, *34*, 2055008.
4. Zhang, L.; Zheng, Y.; Kantoa, R. A review of homomorphic encryption and its applications. In *Proceedings of the 9th EAI International Conference on Mobile Multimedia Communications*; ICST: Brussels, Belgium, 2016; pp. 1–6.
5. Ibtihal, M.; Hassan, N. Homomorphic encryption as a service to outsourced images in mobile cloud computing environment. *Int. J. Cloud Appl. Comput.* **2017**, *7*, 27–40. [CrossRef]
6. Gilad-Bachrach, R.; Dowlin, N.; Laine, K.; Lauter, K.; Naehrig, M.; Wernsing, J. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *International Conference on Machine Learning*; PMLR: Cambridge, MA, USA, 2016; pp. 201–210.
7. Xu, R.; Joshi, J.B.; Li, C. Cryptonn: Training neural networks over encrypted data. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*; IEEE: New York, NY, USA, 2019; pp. 1199–1209.
8. Gentry, C. Fully homomorphic encryption using ideal lattices . In *41st Annual ACM Symposium on Theory of Computing (STOC 2009)*; ACM: New York, NY, USA, 2009; pp. 169–178.
9. Lyubashevsky, V.; Peikert, C.; Regev, O. On ideal lattices and learning with errors over rings. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 1–23.
10. Gentry, C.; Halevi, S.; Smart, N.P. Better bootstrapping in fully homomorphic encryption . In *Proceedings of the 15th International Conference on Practice and Theory in Public-Key Cryptography*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 1–16.
11. Acar, A.; Aksu, H.; Uluagac, A.S.; Conti, M. A survey on homomorphic encryption schemes: Theory and implementation. *ACM Comput. Surv.* **2018**, *51*, 1–35. [CrossRef]
12. Martins, P.; Sousa, L.; Mariano, A. A survey on fully homomorphic encryption: An engineering perspective. *ACM Comput. Surv.* **2017**, *50*, 1–33. [CrossRef]
13. T. V. T. Doan, M. L. Messai, and G. Gavin. A survey on implementations of homomorphic encryption schemes. *Wirel. Pers. Commun.* **2023**, vol. 79, no. 13, pp. 15098–15139. [CrossRef]
14. Marcolla, C.; Sucasas, V.; Manzano, M.; Bassoli, R.; Fitzek, F.H.P.; Aaraj, N. Survey on fully homomorphic encryption, theory, and applications. *Proc. IEEE* **2022**, *110*, 1572–1609. [CrossRef]
15. Kupcova, E.; Voznak, M.; Silhavy, P. A comparative study of partially, somewhat, and fully homomorphic encryption in modern cryptographic libraries. *Comput. Secur.* **2021**, *102*, 102153.
16. Brakerski, Z.; Gentry, C.; Vaikuntanathan, V. (Leveled) fully homomorphic encryption without bootstrapping. *ACM Trans. Comput. Theory* **2014**, *6*, 1–36. [CrossRef]
17. Fan, J.; Vercauteren, F. Somewhat practical fully homomorphic encryption. In *Cryptology ePrint Archive*; International Association for Cryptologic Research (IACR): Bellevue, WA, USA, 2012.
18. Brakerski, Z. Fully homomorphic encryption without modulus switching from classical GapSVP. In *Annual Cryptology Conference*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 868–886.

19. Cheon, J.H.; Kim, A.; Kim, M.; Song, Y. Homomorphic encryption for arithmetic of approximate numbers. In *International Conference on the Theory and Application of Cryptology and Information Security*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 409–437.
20. Gong, Y.; Liu, X.; Li, Z.; Zhang, Y. Practical solutions in fully homomorphic encryption: A survey analyzing existing acceleration methods. *IEEE Access* **2021**, *9*, 105394–105413.
21. Bian, S.; Li, H.; Wang, W. A survey on software-hardware acceleration for fully homomorphic encryption. *J. Electron. Inf. Technol.* **2023**, *45*, 1678–1692.
22. Shen, H.; Zhang, Y.; Wang, J. bootstrapping in approximate fully homomorphic encryption: A research survey. *Cryptography* **2023**, *7*, 23.
23. He, H.N.; Zhang, S.; Liu, L. A survey on fully homomorphic encryption dedicated hardware accelerators. *IEEE Internet Things J.* **2023**, *10*, 10245–10263.
24. Gentry, C. A Fully Homomorphic Encryption Scheme. Ph.D. Thesis, Stanford University, Stanford, CA, USA, 2009.
25. Smart, N.P.; Vercauteren, F. Fully homomorphic encryption with relatively small key and ciphertext sizes. In *International Workshop on Public Key Cryptography*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 420–443.
26. Stehlé, D.; Steinfeld, R. Faster fully homomorphic encryption. In *International Conference on the Theory and Application of Cryptology and Information Security*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 376–392.
27. Gentry, C.; Halevi, S. Fully homomorphic encryption without squashing using depth-3 arithmetic circuits. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*; IEEE: New York, NY, USA, 2011; pp. 107–116.
28. Brakerski, Z.; Vaikuntanathan, V. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In *Annual Cryptology Conference*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 505–524.
29. Brakerski, Z.; Vaikuntanathan, V. Efficient fully homomorphic encryption from (standard) LWE. *SIAM J. Comput.* **2014**, *43*, 831–871. [CrossRef]
30. Brakerski, Z.; Langlois, A.; Peikert, C.; Regev, R.; Stehlé, D. Classical hardness of learning with errors. In *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing*; ACM: New York, NY, USA, 2013; pp. 575–584.
31. Gentry, C.; Halevi, S.; Smart, N.P. Fully homomorphic encryption with polylog overhead. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 250–267.
32. Gomathisankaran, M.; Tyagi, A.; Namuduri, K. HORNS: A homomorphic encryption scheme for cloud computing using residue number system. In *2011 45th Annual Conference on Information Sciences and Systems*; IEEE: New York, NY, USA, 2011; pp. 1–5.
33. Bajard, J.-C.; Eynard, J.; Hasan, M.A.; Zucca, V. A full RNS variant of FV like somewhat homomorphic encryption schemes. In *International Conference on Selected Areas in Cryptography*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 423–442.
34. Halevi, S.; Polyakov, Y.; Shoup, V. An improved RNS variant of the BFV homomorphic encryption scheme. In *Cryptographers' Track at the RSA Conference*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 83–105.
35. Cheon, J.H.; Han, K.; Kim, A.; Kim, M.; Song, Y. A full RNS variant of approximate homomorphic encryption. In *International Conference on Selected Areas in Cryptography* Springer: Berlin/Heidelberg, Germany, 2018; pp. 345–374.
36. Han, K.; Ki, D. Better bootstrapping for approximate homomorphic encryption. In *Cryptographers' Track at the RSA Conference*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 221–245.
37. Ma, S.; Huang, T.; Wang, A.; Wang, H. Accelerating BGV bootstrapping for large p using null polynomials over $\mathbb{Z}_{p^e}$. In *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques*; Springer: Berlin/Heidelberg, Germany, 2024; pp. 357–386.
38. Smart, N.P.; Vercauteren, F. Fully homomorphic SIMD operations. *Des. Codes Cryptogr.* **2014**, *71*, 57–81.
39. Yasuda, M.; Shimoyama, T.; Kogure, J.; Koshiba, T.; Yokoyama, K. New packing method in somewhat homomorphic encryption and its applications. *Secur. Commun. Netw.* **2015**, *8*, 2194–2213. [CrossRef]
40. Yin, X.K. Application of the Chinese remainder theorem in polynomial multiplication calculation. *J. Hebei Acad. Sci.* **2012**, *29*, 5–9.
41. Gentry, C.; Sahai, A.; Waters, B. Homomorphic encryption from learning with errors: Conceptually simpler, asymptotically faster, attribute-based. In *Annual Cryptology Conference*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 75–92.
42. Alperin-Sheriff, J.; Peikert, C. Faster bootstrapping with polynomial error. In *Annual Cryptology Conference*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 297–314.
43. Ducas, L.; Micciancio, D. FHEW: Bootstrapping homomorphic encryption in less than a second. In *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques*; Springer: Berlin/Heidelberg, Germany, 2015; pp. 617–640.
44. Chillotti, I.; Gama, N.; Georgieva, M.; Izabachène, M. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In *International Conference on the Theory and Application of Cryptology and Information Security*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 3–33.

45. Chillotti, I.; Gama, N.; Georgieva, M.; Izabachène, M. Faster packed homomorphic operations and efficient circuit bootstrapping for TFHE. In *International Conference on the Theory and Application of Cryptology and Information Security*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 379–408.
46. Espitau, T.; Joux, A.; Kharchenko, N. On a hybrid approach to solve binary-LWE. In *IACR Cryptology ePrint Archive*; International Association for Cryptologic Research (IACR): Bellevue, WA, USA, 2020.
47. Micciancio, D.; Polyakov, Y. Bootstrapping in FHEW-like cryptosystems. In *Proceedings of the 9th on Workshop on Encrypted Computing & Applied Homomorphic Cryptography*; ACM: New York, NY, USA, 2021, pp. 1–12.
48. Bae, Y.; Cheon, J.H.; Cho, W.; Kim, J.; Lee, J. Meta-bts: Bootstrapping precision beyond the limit. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*; ACM: New York, NY, USA, 2022; pp. 2491–2505.
49. Bonte, C.; Iliashenko, I.; Park, J.; Vercauteren, F. FINAL: Faster FHE instantiated with NTRU and LWE. In *International Conference on the Theory and Application of Cryptology and Information Security*; Springer: Berlin/Heidelberg, Germany, 2022; pp. 559–589.
50. Lee, C.; Min, S.; Seo, J.; Cheon, J.H. Faster TFHE bootstrapping with block binary keys. In *Proceedings of the 2023 ACM Asia Conference on Computer and Communications Security*; ACM: New York, NY, USA, 2023; pp. 15–26.
51. Xiang, B.; Zhang, J.; Deng, Y.; Zhang, Z. Fast blind rotation for bootstrapping FHEs. In *Annual International Cryptology Conference*; Springer: Berlin/Heidelberg, Germany, 2023; pp. 289–318.
52. Wang, R.; Wen, Y.; Li, Z.; Zhang, X. Circuit bootstrapping: Faster and smaller. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*; Springer: Berlin/Heidelberg, Germany, 2024; pp. 183–212.
53. Lee, Y.; Micciancio, D.; Kim, A.; Cheon, J.H. Efficient FHEW bootstrapping with small evaluation keys, and applications to threshold homomorphic encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*; Springer: Berlin/Heidelberg, Germany, 2023; pp. 643–673.
54. Zhao, X.F.; Wu, M.; Song, W.T. bootstrapping optimization technology for FINAL FHE scheme. *J. Electron. Inf. Technol.* **2025**, *47*, 7.
55. Wang, W.; Huang, X. FPGA implementation of a large-number multiplier for fully homomorphic encryption. In *2013 IEEE International Symposium on Circuits and Systems (ISCAS)*; IEEE: New York, NY, USA, 2013; pp. 2861–2864.
56. Doröz, Y.; Öztürk, E.; Sunar, B. Accelerating Fully Homomorphic Encryption in Hardware. *IEEE Trans. Comput.* **2014**, *64*, 1509–1521. [CrossRef]
57. Moore, C.; Hanley, N.; McAllister, J.; O'Neill, M. Targeting FPGA DSP slices for a large integer multiplier for integer based FHE. In *International Conference on Financial Cryptography and Data Security*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 112–123.
58. Pöppelmann, T.; Naehrig, M.; Putnam, A.; Rohloff, K. Accelerating homomorphic evaluation on reconfigurable hardware. In *International Workshop on Cryptographic Hardware and Embedded Systems*; Springer: Berlin/Heidelberg, Germany, 2015; pp. 307–326.
59. Roy, S.S.; Järvinen, K.; Vercauteren, F.; Verbauwhede, I. Modular hardware architecture for somewhat homomorphic function evaluation. In *International Workshop on Cryptographic Hardware and Embedded Systems*; Springer: Berlin/Heidelberg, Germany, 2015; pp. 164–184.
60. Öztürk, E.; Doröz, Y.; Savaş, E.; Sunar, B. Accelerating somewhat homomorphic evaluation using FPGAs. In *IACR Cryptology ePrint Archive*; International Association for Cryptologic Research (IACR): Bellevue, WA, USA, 2015.
61. Cousins, D.B.; Rohloff, K.; Sumorok, D. Designing an FPGA-accelerated homomorphic encryption co-processor. *IEEE Trans. Emerg. Top. Comput.* **2016**, *5*, 193–206. [CrossRef]
62. Roy, S.S.; Järvinen, K.; Vliegen, J.; Verbauwhede, I. HEPCloud: An FPGA-based multicore processor for FV somewhat homomorphic function evaluation. *IEEE Trans. Comput.* **2018**, *67*, 1637–1650.
63. Mert, A.C.; Öztürk, E.; Savaş, E. Design and implementation of encryption/decryption architectures for BFV homomorphic encryption scheme. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2019**, *28*, 353–362. [CrossRef]
64. Turan, F.; Roy, S.S.; Verbauwhede, I. HEAWS: An accelerator for homomorphic encryption on the Amazon AWS FPGA. *IEEE Trans. Comput.* **2020**, *69*, 1185–1196.
65. Riazi, M.S.; Laine, K.; Pelton, B.; Dai, W. HEAX: An architecture for computing on encrypted data. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*; ACM: New York, NY, USA, 2020; pp. 565–579.
66. Roy, S.S.; Mert, A.C.; Kwon, S.; Yoo, D. Accelerator for computing on encrypted data. In *IACR Cryptology ePrint Archive*; International Association for Cryptologic Research (IACR): Bellevue, WA, USA, 2021.
67. Samardzic, N.; Feldmann, A.; Krastev, A.; Dubrovina, M. F1: A fast and programmable accelerator for fully homomorphic encryption. In *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*; ACM: New York, NY, USA, 2021; pp. 1082–1095.
68. Samardzic, N.; Feldmann, A.; Krastev, A.; Dubrovina, M. Craterlake: A hardware accelerator for efficient unbounded computation on encrypted data. In *Proceedings of the 49th Annual International Symposium on Computer Architecture*; ACM: New York, NY, USA, 2022; pp. 167–181.
69. Mert, A.C.; Kwon, S.; Shin, Y.; Yoo, D.; Feldmann, A. Medha: Microcoded hardware accelerator for computing on encrypted data. *arXiv* **2022**, arXiv:2210.05476.

70. Kim, J.; Lee, G.; Kim, S.; Huh, J. Ark: Fully homomorphic encryption accelerator with runtime data generation and inter-operation key reuse. In *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*; IEEE: New York, NY, USA, 2022; pp. 1156–1169.
71. Yang, Y.; Zhang, H.; Fan, S.; Liu, L. Poseidon: Practical homomorphic encryption accelerator. In *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*; IEEE: New York, NY, USA, 2023; pp. 721–734.
72. Bertels, J. Fastest homomorphic encryption in the West made faster. *arXiv* 2024. Available online: https://cosicdatabase.esat.kuleuven.be/backend/publications/files/these/423 (accessed on 2 June 2026).
73. Jiang, L.; Lou, Q.; Joshi, N. Matcha: A fast and energy-efficient accelerator for fully homomorphic encryption over the torus. In *Proceedings of the 59th ACM/IEEE Design Automation Conference*; ACM: New York, NY, USA, 2022; pp. 961–966.
74. Beirendonck, M.V.; D'Anvers, J.-P.; Turan, F.; Verbauwhede, I. FPT: A fixed-point accelerator for torus fully homomorphic encryption. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*; ACM: New York, NY, USA, 2023; pp. 1873–1886.
75. Zhang, K.; Wang, A.; Parhi, K.K.; Zhao, C. Hardware Acceleration for Fully Homomorphic Encryption Scheme Switching from CKKS to FHEW. In *2024 58th Asilomar Conference on Signals, Systems, and Computers*; IEEE: New York, NY, USA, 2024; pp. 1–6.
76. Gentry, C.; Halevi, S. Implementing Gentry's fully-homomorphic encryption scheme. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 129–148.
77. Millar, K.; Łukowiak, M.; Radziszowski, S. Design of a flexible Schönhage-Strassen FFT polynomial multiplier with high-level synthesis to accelerate HE in the cloud. In *2019 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*; IEEE: New York, NY, USA, 2019; pp. 1–8.
78. Coron, J.-S.; Mandal, A.; Naccache, D.; Tibouchi, M. Fully homomorphic encryption over the integers with shorter public keys. In *Annual Cryptology Conference*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 487–504.
79. Antognazza, F.; Barenghi, A.; Pelosi, G.; Wang, Y. A flexible ASIC-oriented design for a full NTRU accelerator. In *Proceedings of the 28th Asia and South Pacific Design Automation Conference*; ACM: New York, NY, USA, 2023; pp. 424–429.
80. Moore, C.; O'Neill, M.; Hanley, N.; McAllister, J. Accelerating integer-based fully homomorphic encryption using Comba multiplication. In *2014 IEEE Workshop on Signal Processing Systems (SiPS)*; IEEE: New York, NY, USA, 2014; pp. 1–6.
81. Bos, J.W.; Lauter, K.; Loftus, J.; Naehrig, M. Improved security for a ring-based fully homomorphic encryption scheme. In *IMA International Conference on Cryptography and Coding*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 45–64.
82. Sheikhpour, S.; Sadi, M.H.; Mahani, A. High-throughput configurable SIMON architecture for flexible security. *Microelectron. J.* **2021**, *113*, 105085. [CrossRef]
83. López-Alt, A.; Tromer, E.; Vaikuntanathan, V. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *Proceedings of the Forty-Fourth Annual ACM Symposium on Theory of Computing*; ACM: New York, NY, USA, 2012; pp. 113–122.
84. Alkady, Y.; Farouk, F.; Rizk, R. Fully homomorphic encryption with AES in cloud computing security. In *International Conference on Advanced Intelligent Systems and Informatics*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 325–334.
85. Öztürk, E.; Doröz, Y.; Savaş, E.; Sunar, B. A custom accelerator for homomorphic encryption applications. *IEEE Trans. Comput.* **2016**, *66*, 3–16. [CrossRef]
86. Liu, S.; Hu, B. Homomorphic evaluation of block cipher circuits based on MGSW15 scheme. *J. Sichuan Univ. Nat. Sci. Ed.* **2019**, *56*, 661–670.
87. Xia, K.X. Implementation of Subfield Attack Algorithm for NTRU Problem and Number Theoretic Transform. Master's Thesis, Shandong University, Jinan, China, 2017.
88. Roy, S.S.; Turan, F.; Järvinen, K.; Verbauwhede, I. FPGA-based high-performance parallel architecture for homomorphic computing on encrypted data. In *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*; IEEE: New York, NY, USA, 2019; pp. 318–329.
89. Fawaz, S.M.; Belal, N.; Elrefaey, A.; Abdelrahman, M. A comparative study of homomorphic encryption schemes using microsoft seal. *J. Phys. Conf. Ser.* **2021**, *2128*, 012045. [CrossRef]
90. Lee, J.; Duong, P.N.; Lee, H. Configurable encryption and decryption architectures for CKKS-based homomorphic encryption. *Sensors* **2023**, *23*, 7389. [CrossRef] [PubMed]
91. Bae, Y.; Cheon, J.H.; Kim, J.; Lee, J. HERMES: Efficient ring packing using MLWE ciphertexts and application to transciphering. In *Annual International Cryptology Conference*; Springer: Berlin/Heidelberg, Germany, 2023; pp. 489–518.
92. Soni, D.; Neda, N.; Zhang, N.; Patra, S.K. RPU: The ring processing unit. In *2023 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*; IEEE: New York, NY, USA, 2023; pp. 1–10.
93. Dai, W.; Sunar, B. cuHE: A Homomorphic Encryption Accelerator Library. In *International Conference on Cryptography and Information Security in the Balkans*; Springer: Berlin/Heidelberg, Germany, 2015.
94. Chan, Y.H.; Yang, H.; Shen, S.; Fan, X.; Lyu, S.; Hung, P.S.Y.; Cheung, R.C.C. HHEML: Hybrid Homomorphic Encryption for Privacy-Preserving Machine Learning on Edge. *arXiv* **2025**, arXiv:2510.20243.

95. Aikata, A.; Sanz Sobrino, D.; Sinha Roy, S. Pasta on Edge: Cryptoprocessor for Hybrid Homomorphic Encryption. In *2025 Design, Automation & Test in Europe Conference (DATE)*; IEEE: New York, NY, USA, 2025.
96. Di Matteo, S.; Gerfo, M.L.; Saponara, S. VLSI Design and FPGA Implementation of an NTT Hardware Accelerator for Homomorphic SEAL-Embedded Library. *IEEE Access* **2023**, 11, 72498–72508. [CrossRef]
97. Guerrini, V.; Sorrentino, G.; Barenghi, A.; Conficconi, D. ReFHE-NTT: Resource-Driven NTT FPGA Architecture for Fully Homomorphic Encryption. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.* **2024**, *43*, 2290–2303.
98. Krieger, F.; Hirner, F.; Mert, A.C. Aloha-he: A Low-Area Hardware Accelerator for Client-Side Operations in Homomorphic Encryption. In *2024 Design, Automation & Test in Europe Conference & Exhibition (DATE)*; IEEE: New York, NJ, USA, 2024; pp. 1–6.
99. Meng, X.; Lyu, Y. FedBit: Accelerating Privacy-Preserving Federated Learning via Bit-Interleaved Packing and Cross-Layer Co-Design. *arXiv* **2025**, arXiv:2509.23091.
100. Yang, Z.; Hu, S.; Chen, K. FPGA-Based Hardware Accelerator of Homomorphic Encryption for Efficient Federated Learning. *arXiv* **2020**, arXiv:2007.10560.
101. Chen, Z.; Gu, Z.; Lu, Y.; Ren, X.; Zhong, R.; Lu, W.J.; Zhang, J.; Zhang, Y.; Wu, H.; Zheng, X.; et al. SAFE: A Scalable Homomorphic Encryption Accelerator for Vertical Federated Learning. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.* **2025**, *44*, 1662–1675. [CrossRef]
102. Ren, X.; Chen, H.; Liu, B.; Zhang, C.; Dai, H.; Liu, L. CHAM: A Customized Homomorphic Encryption Accelerator for Fast Matrix-Vector Product. In *2023 60th ACM/IEEE Design Automation Conference (DAC)*; IEEE: New York, NY, USA, 2023; pp. 1–6.
103. Jiang, M.; Zhu, Y.; You, H.; Tan, C.; Li, Z.; Xu, J.; Ju, L. FHE-CGRA: Enable Efficient Acceleration of Fully Homomorphic Encryption on CGRAs. In *Proceedings of the 61st ACM/IEEE Design Automation Conference*; ACM: New York, NY, USA, 2024; pp. 1–6.
104. Reagen, B.; Choi, W.; Ko, Y.; Lee, V.; Wei, G.-Y.; Lee, H.-H.S.; Brooks, D. Cheetah: Optimizing and Accelerating Homomorphic Encryption for Private Inference. In *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*; IEEE: New York, NY, USA, 2021; pp. 26–39.
105. Golder, A.; Kumar, R.; Taneja, S.; Race, K.; Aseron, P.; Greensky, J.; Wang, W.; Gong, H.; Kethareswaran, L.; Suresh, V.; et al. HERACLES: 8192-Way SIMD Programmable Scalable Fully-Homomorphic Encryption SoC for Privacy-Preserving Cloud Computing in Intel 3 CMOS. In *2026 IEEE International Solid-State Circuits Conference (ISSCC)*; IEEE: New York, NY, USA, 2026; pp. 424–426.
106. Langlois, A.; Stehlé, D. Hardness of decision (R)LWE for any modulus. IACR Cryptol. In *IACR Cryptology ePrint Archive*; International Association for Cryptologic Research (IACR): Bellevue, WA, USA, 2012.
107. Lips, B. The Efficiency of Polynomial Multiplication Methods for Ring-Based PQC Algorithms of Round 3 of the NIST PQC Competition. Master's Thesis, Eindhoven University of Technology, Eindhoven, The Netherlands, 2021.
108. Pöppelmann, T.; Güneysu, T. Towards efficient arithmetic for lattice-based cryptography on reconfigurable hardware. In *International Conference on Cryptology and Information Security in Latin America*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 145–160.
109. Xu, C.; Yu, H.; Xi, W.; Zhou, Y. A polynomial multiplication accelerator for faster lattice cipher algorithm in security chip. *Electronics* **2023**, *12*, 951. [CrossRef]
110. Roy, S.S.; Vercauteren, F.; Mentens, N. I. Verbauwhede, Compact ring-LWE cryptoprocessor. In *International Workshop on Cryptographic Hardware and Embedded Systems*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 371–387.
111. Du, C.; Bai, G. Towards efficient polynomial multiplication for lattice-based cryptography. In *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*; IEEE: New York, NY, USA, 2016; pp. 1162–1165.
112. Pöppelmann, T.; Oder, T.; Güneysu, T. High-performance ideal lattice-based cryptography on 8-bit ATxmega microcontrollers. In *International Conference on Cryptology and Information Security in Latin America*; Springer: Berlin/Heidelberg, Germany, 2015; pp. 245–264.
113. Zhang, N.; Yang, B.; Chen, C.; Wang, X. Highly efficient architecture of NewHope-NIST on FPGA using low-complexity NTT/INTT. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* **2020**, *2020*, 49–72. [CrossRef]
114. Xin, G.; Han, J.; Yin, T.; Liu, X. VPQC: A domain-specific vector processor for post-quantum cryptography based on RISC-V architecture. *IEEE Trans. Circuits Syst. I Reg. Pap.* **2020**, *67*, 2672–2684. [CrossRef]
115. Chen, D.D.; Mentens, N.; Vercauteren, F.; Verbauwhede, I. High-speed polynomial multiplication architecture for ring-LWE and SHE cryptosystems. *IEEE Trans. Circuits Syst. I Reg. Pap.* **2014**, *62*, 157–167.
116. Banerjee, U.; Ukyab, T.S.; Chandrakasan, A.P. Sapphire: A configurable crypto-processor for post-quantum lattice-based protocols. *arXiv* **2019**, arXiv:1910.07557.
117. Longa, P.; Naehrig, M. Speeding up the number theoretic transform for faster ideal lattice-based cryptography. In *International Conference on Cryptology and Network Security*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 124–139.
118. Duong-Ngoc, P.; Kwon, S.; Yoo, D.; Lee, H. Area-efficient number theoretic transform architecture for homomorphic encryption. *IEEE Trans. Circuits Syst. I Reg. Pap.* **2022**, *70*, 1270–1283. [CrossRef]

119. Mandal, S.; Roy, D.B. Winograd for NTT: A case study on higher-radix and low-latency implementation of NTT for post quantum cryptography on FPGA. *IEEE Trans. Circuits Syst. I Reg. Pap.* **2024**, *71*, 6396–6409. [CrossRef]
120. Ramyavani, P.; Casheekar, A.; Reddy, T.; Kumar, S. Accelerating polynomial multiplication using a three-stage NTT with radix-2 butterfly unit. In *2025 International VLSI Symposium on Technology, Systems and Applications (VLSI TSA)*; IEEE: New York, NY, USA, 2025; pp. 1–5.
121. Taghavi, B.; Azarderakhsh, R.; Mozaffari Kermani, M. ParallelNTT: Maximizing performance of forward and inverse NTT on FPGA for ML-DSA and ML-KEM. In *Proceedings of the Great Lakes Symposium on VLSI 2025*; ACM: New York, NY, USA, 2025; pp. 321–326.
122. Zhao, Y.; Xie, Z.; Zhan, R.; Wang, L. NTT-LSU: Tightly coupled architecture for efficient NTT implementation on RISC-V processor. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2025**, *45*, 2952–2961.
123. Fritzmann, T.; Sepúlveda, J. Efficient and flexible low-power NTT for lattice-based cryptography. In *2019 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*; IEEE: New York, NY, USA, 2019; pp. 102–107.
124. Mert, A.C.; Karabulut, E.; Öztürk, E.; Savaş, E. An extensive study of flexible design methods for number theoretic transform. *IEEE Trans. Comput.* **2020**, *71*, 2829–2843. [CrossRef]
125. Mert, A.C.; Öztürk, E.; Savaş, E. FPGA implementation of a run-time configurable NTT-based polynomial multiplication hardware. *Microprocess. Microsyst.* **2020**, *78*, 103219. [CrossRef]
126. Hu, X.; Tian, J.; Li, M.; Wang, Z. AC-PM: An area-efficient and configurable polynomial multiplier for lattice based cryptography. *IEEE Trans. Circuits Syst. I Reg. Pap.* **2022**, *70*, 719–732.
127. Derya, K.; Mert, A.C.; Öztürk, E.; Savaş, E. CoHA-NTT: A configurable hardware accelerator for NTT-based polynomial multiplication. *Microprocess. Microsyst.* **2022**, *89*, 104451. [CrossRef]
128. Geng, Y.; Hu, X.; Li, M.; Wang, Z. Rethinking parallel memory access pattern in number theoretic transform design. *IEEE Trans. Circuits Syst. II Express Briefs* **2023**, *70*, 1689–1693. [CrossRef]
129. Tan, W.; Au, A.; Aase, B.; Güneysu, T. An efficient polynomial multiplier architecture for the bootstrapping algorithm in a fully homomorphic encryption scheme. In *2019 IEEE International Workshop on Signal Processing Systems (SiPS)*; IEEE: New York, NY, USA, 2019; pp. 220–225.
130. Yaman, F.; Mert, A.C.; Öztürk, E.; Savaş, E. A hardware accelerator for polynomial multiplication operation of CRYSTALS-KYBER PQC scheme. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*; IEEE: New York, NY, USA, 2021; pp. 1–6.
131. Migliore, V.; Real, M.M.; Lapotre, V.; Tisserand, A. Exploration of polynomial multiplication algorithms for homomorphic encryption schemes. In *2015 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*; IEEE: New York, NY, USA, 2015; pp. 1–6.
132. Zhu, Y.; Zhu, M.; Yang, B.; Wang, X. A high-performance hardware implementation of Saber based on Karatsuba algorithm. In *IACR Cryptology ePrint Archive*; International Association for Cryptologic Research (IACR): Bellevue, WA, USA, 2020.
133. Wong, Z.-Y.; Wong, D.C.-K.; Lee, W.-K.; Liu, C.-W. High-speed RLWE-oriented polynomial multiplier utilizing Karatsuba algorithm. *IEEE Trans. Circuits Syst. II Express Briefs* **2021**, *68*, 2157–2161. [CrossRef]
134. Wang, J.; Yang, C.; Zhang, F.; Chen, S. TCPM: A reconfigurable and efficient Toom-Cook-based polynomial multiplier over rings using a novel compressed postprocessing algorithm. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2023**, *31*, 1153–1166. [CrossRef]
135. Mera, J.M.B.; Karmakar, A.; Roy, S.S.; Verbauwhede, I. Saber on ARM: CCA-secure module lattice-based key encapsulation on ARM. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* **2018**, *2018*, 243–266. [CrossRef]
136. Mera, J.M.B.; Karmakar, A.; Verbauwhede, I. Time-memory trade-off in Toom-Cook multiplication: An application to module-lattice based cryptography. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* **2020**, *2020*, 222–244. [CrossRef]
137. Kannwischer, M.J.; Rijneveld, J.; Schwabe, P. Faster multiplication on Cortex-M4 to speed up NIST PQC candidates. In *International Conference on Applied Cryptography and Network Security*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 168–185.
138. Paksoy, İ.K.; Cenk, M. Faster NTRU on ARM Cortex-M4 with TMVP-based multiplication. *IEEE Trans. Circuits Syst. I Reg. Pap.* **2022**, *69*, 4083–4092. [CrossRef]
139. Mera, J.M.B.; Turan, F.; Karmakar, A.; Verbauwhede, I. Compact domain-specific co-processor for accelerating module lattice-based KEM. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*; IEEE: New York, NY, USA, 2020; pp. 1–6.
140. Ghosh, A.; Mera, J.M.B.; Karmakar, A.; Verbauwhede, I. A 334μW 0.158 mm^2 Saber learning with rounding based post-quantum crypto accelerator. In *2022 IEEE Custom Integrated Circuits Conference (CICC)*; IEEE: New York, NY, USA, 2022; pp. 1–4.
141. Gu, Z.; Li, S. A division-free Toom–Cook multiplication-based Montgomery modular multiplication. *IEEE Trans. Circuits Syst. II Express Briefs* **2018**, *66*, 1401–1405.
142. Ding, J.; Li, S.; Gu, Z. High-speed ECC processor over NIST prime fields applied with Toom–Cook multiplication. *IEEE Trans. Circuits Syst. I Reg. Pap.* **2018**, *66*, 1003–1016.
143. Ma, Y.; Ding, H.Y.; Li, Z.C. FPGA fast implementation of SNTRUP algorithm based on Toom-Cook polynomial multiplication. *Softw. Eng. Appl.* **2023**, *12*, 402.

144. Vasanthi, D.; Reddy, S.G.K.; Rao, M. HRM: M-term heterogeneous hybrid blend recursive multiplier for GF(2^n) polynomial. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2024**, *32*, 1447–1460. [CrossRef]
145. Das, M.; Jajodia, B. FPGA implementation of hybrid Karatsuba multiplications for NIST post-quantum cryptographic hardware primitives. In *2022 19th International SoC Design Conference (ISOCC)*; IEEE: New York, NY, USA, 2022; pp. 229–233.
146. Roche, D.S. Space-and time-efficient polynomial multiplication. In *Proceedings of the 2009 International Symposium on Symbolic and Algebraic Computation*; ACM: New York, NY, USA, 2009; pp. 351–358.
147. D'Anvers, J.-P.; Karmakar, A.; Roy, S.S.; Vercauteren, F. Saber: Module-LWR based key exchange, CPA-secure encryption and CCA-secure KEM. In *International Conference on Cryptology in Africa*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 287–306.
148. Hao, Y.; Wang, W.; Dang, H.; Zhao, X. Efficient Barrett modular multiplication based on Toom-Cook multiplication. *IEEE Trans. Circuits Syst. II Express Briefs* **2023**, *71*, 862–866. [CrossRef]
149. Shoup, V. NTL: A Library for Doing Number Theory. 2001. Available online: https://www.iaa.tu-bs.de/AppliedAlgebra/ICMS2020/ICMS2020_files/Shoup_abstract.pdf (accessed on 2 June 2026).
150. Kim, S.; Lee, K.; Cho, W.; Choi, S. FPGA-based accelerators of fully pipelined modular multipliers for homomorphic encryption. In *2019 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*; IEEE: New York, NY, USA, 2019; pp. 1–6.
151. Mazonka, O.; Moopan, M.N.T.; Maniatakos, M. Exploring generalization of Shoup modular multiplier. In *Proceedings of the Great Lakes Symposium on VLSI 2024*; ACM: New York, NY, USA, 2024; pp. 216–221.
152. Li, H.; Ren, S.; Wang, W.; Chen, Y. A low-cost high-performance Montgomery modular multiplier based on pipeline interleaving for IoT devices. *Electronics* **2023**, *12*, 3241. [CrossRef]
153. Roy, A.; Maharana, R.; Singh, S.K.; Acharya, D.P. Improved Montgomery modular multipliers on FPGAs and ASICs. *IEEE Embed. Syst. Lett.* **2025**, *18*, 208–211. [CrossRef]
154. Wang, J.; Wang, X.; Liu, W.; Zhang, J. A parallel and pipelined high speed Montgomery modular multiplier for IoT devices. *Comput. Netw.* **2025**, *265*, 111282. [CrossRef]
155. Cao, X.; Moore, C.; O'Neill, M.; McAllister, J. Optimised multiplication architectures for accelerating FHE. *IEEE Trans. Comput.* **2015**, *65*, 2794–2806.
156. Zijlstra, T.; Bigou, K.; Tisserand, A. FPGA implementation and comparison of protections against SCAs for RLWE. In *International Conference on Cryptology in India*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 207–226.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.