Journal of Electronic Research and Application, 2025, Volume 9, Issue 6
http://ojs.bbwpublisher.com/index.php/JERA
ISSN Online: 2208-3510
ISSN Print: 2208-3502

Secure and Privacy-Preserving Cross-Departmental Computation Framework Based on BFV and Blockchain

Peng Zhao, Yu Du

College of Computer Science and Technology, Taiyuan Normal University, Jinzhong 030619, China

**Author to whom correspondence should be addressed.*

Abstract: As the demand for cross-departmental data collaboration continues to grow, traditional encryption methods struggle to balance data privacy with computational efficiency. This paper proposes a cross-departmental privacy-preserving computation framework based on BFV homomorphic encryption, threshold decryption, and blockchain technology. The proposed scheme leverages homomorphic encryption to enable secure computations between sales, finance, and taxation departments, ensuring that sensitive data remains encrypted throughout the entire process. A threshold decryption mechanism is employed to prevent single-point data leakage, while blockchain and IPFS are integrated to ensure verifiability and tamper-proof storage of computation results. Experimental results demonstrate that with 5,000 sample data entries, the framework performs efficiently and is highly scalable in key stages such as sales encryption, cost calculation, and tax assessment, thereby validating its practical feasibility and security.

Keywords: Homomorphic encryption; Zero-knowledge proof; Blockchain; Cross-departmental privacy-preserving computation

Online publication: December 16, 2025

1. Introduction

With growing concerns over data privacy, a pressing challenge has emerged: how to enable collaborative computation across departments without exposing sensitive data [1]. This issue is particularly critical in sectors such as finance, healthcare, and government, where multiple departments must share and jointly analyze sensitive, multi-source information. Such scenarios demand higher standards of confidentiality and regulatory compliance [2]. While traditional symmetric and asymmetric encryption methods can ensure secure data transmission, they typically require decryption before computation, which introduces potential data leakage risks and reduces computational efficiency [3].

Homomorphic encryption (HE) allows computations to be performed directly on encrypted data, ensuring that

information remains protected throughout the entire processing workflow. This capability significantly enhances privacy and offers a systematic approach to privacy-preserving computation [4]. Meanwhile, blockchain technology, with its immutability and traceability, provides a reliable foundation for secure data provenance and auditability [5]. Additionally, threshold cryptography strengthens system security through distributed key management, preventing single points of failure [6]. In cross-department collaborative computing scenarios, combining homomorphic encryption with blockchain technology ensures that sensitive data remains encrypted and verifiable throughout its entire lifecycle. This not only prevents data breaches but also supports future compliance audits [7]. As business datasets continue to grow in size, ensuring secure and efficient large-scale homomorphic computation has become a key research challenge.

In this context, this paper explores how homomorphic encryption can be applied to cross-department data computation. Specifically, it investigates the use of the Brakerski-Fan-Vercauteren (BFV) homomorphic encryption scheme to enable joint analysis of sales, cost, and tax data. We propose a multi-stage computational framework that allows departments to securely perform addition and multiplication operations while keeping data encrypted, thus enabling efficient and secure collaborative financial analysis. The main contributions of this paper are as follows:

(1) We propose a homomorphic encryption-based collaboration framework tailored to multi-department financial workflows. Under unified parameters and batching encodings, it supports end-to-end privacy protection for sales aggregation, cost consolidation, and tax calculation;
(2) We design a collaborative mechanism that integrates threshold decryption and blockchain-based notarization. Security is enhanced through private key sharding and joint decryption, while metadata is recorded on-chain to support traceable audits;
(3) We conduct a systematic performance evaluation, analyzing latency and resource overhead for encryption, decryption, rotation summation, and plaintext multiplication on datasets of equal scale, providing experimental insights for future optimization.

2. Related work

In the domain of verifiable privacy-preserving transactions combining HE with Zero-Knowledge Proofs (ZKPs), literature presents a scheme based on additive homomorphic encryption systems such as Paillier [8]. By introducing Fujisaki-Okamoto commitments alongside zero-knowledge proofs, the scheme not only encrypts transaction data but also enables zero-knowledge validation of transaction amounts [9]. However, this approach increases overall proof complexity, which in turn limits system efficiency. To address the computational bottlenecks of Paillier's additive homomorphism, literature proposes an enhanced Okamoto-Uchiyama (OU) homomorphic encryption scheme [10]. It incorporates off-chain payment channels to reduce on-chain transaction frequency, striking a better balance between privacy protection and scalability. This approach proves effective in decentralized, open environments, improving throughput and supporting large-scale privacy-preserving applications [11].

Another notable category of encryption schemes is the hybrid "PailGamal" system, which combines Paillier and ElGamal encryption. As discussed in literature, this system is paired with Bulletproofs range proofs to enhance the efficiency of privacy protection [12]. While it achieves a solid balance between privacy and verifiability, it also introduces the risk of regulators holding the system's private keys. Furthermore, it continues to face scalability limitations in decentralized settings. To further improve scalability and privacy performance, literature introduces

a novel encryption scheme based on hierarchical multi-party key management [13]. By integrating threshold encryption and scalable ZKP protocols, the system offers robust privacy protection while supporting efficient verification and oversight in multi-party collaborative environments.

In addition, threshold encryption and partial decryption mechanisms have seen widespread adoption in privacy protection and regulatory compliance scenarios. Literature proposes a blockchain-based privacy-preserving scheme utilizing threshold key mechanisms to distribute encryption and decryption tasks [14]. Data decryption is permitted only when predefined threshold conditions are met, which effectively eliminates the risk of single points of failure, an advantage well-suited to cross-department applications requiring compliance oversight. While this approach strikes a good balance between privacy and compliance, practical challenges remain, including the efficient management of key shares and ensuring the integrity of data transmission [15].

Literature focuses on integrating threshold homomorphic encryption with compliance oversight [7]. It proposes a threshold homomorphic encryption architecture that allows multiple parties to collaboratively decrypt encrypted transaction data. This solution preserves data privacy while meeting regulatory requirements. By implementing fine-grained access control for participating entities, the system ensures privacy protection and enables regulatory verification of blockchain data, addressing the longstanding issue of limited regulatory intervention in traditional privacy-preserving systems.

Nevertheless, despite achieving a degree of balance between privacy and verifiability, existing schemes still face critical challenges. In particular, within open and decentralized environments, it remains difficult to ensure data privacy while maintaining system scalability and enabling practical regulatory oversight [16].

In summary, although blockchain-based privacy-preserving schemes that combine homomorphic encryption with zero-knowledge proofs have partially addressed the trade-off between privacy and verifiability, limitations persist in terms of efficiency, scalability, and regulatory compliance. Building upon these foundations, this study proposes a novel privacy-preserving framework that integrates threshold homomorphic encryption and zero-knowledge proofs. The goal is to enhance system scalability and compliance while ensuring strong privacy guarantees, thereby offering a more practical and applicable solution for privacy protection in blockchain-based systems.

3. Related background

3.1. Homomorphic encryption and the BFV scheme

Homomorphic encryption enables computations to be performed directly on ciphertexts such that, after decryption, the resulting plaintext matches the outcome of the same operations executed on the original data. Schemes that support both homomorphic addition and multiplication can evaluate arbitrary polynomial functions. The BFV scheme is a representative lattice-based homomorphic encryption construction grounded in the Ring Learning With Errors (Ring-LWE) assumption. Let R be a polynomial ring, and (q,t) be the modulus pair, where q is the ciphertext modulus and t is the plaintext modulus. Key generation, encryption, and decryption are performed over R and its associated modular rings.

3.1.1. Key generation

Let the secret vector be $s \in R$, and sample random values $u, e_1, e_2 \leftarrow x$. The public key is:

$$pk = (b, a) = (-(a \cdot s + e_1), a) \quad (1)$$

3.1.2. Encryption

Given a plaintext message $m \in R_1$, randomly sample $u, e_1, e_2 \leftarrow x$, and output the ciphertext:

$$c = (c_0, c_1) = (bu + e_1 + \left\lfloor \frac{q}{t} \right\rfloor m, au + e_2) \in R_q^2 \quad (2)$$

3.1.3. Decryption

Compute $v = c_0 + c_1 s \in R_q$, $\widetilde{m} = \left\lfloor \frac{t}{q} \cdot v \right\rceil mod t \in R_t$, where $\lfloor \cdot \rceil$ denotes rounding to the nearest integer.

3.1.4. Homomorphic operations

Addition is performed component-wise:

$$(c_0, c_1) + (c_{0'}, c_{1'}) = (c_0 + c_{0'}, c_1 + c_{1'}) \quad (3)$$

3.1.5. Multiplication

Ciphertext multiplication results in a ciphertext triplet and causes an increase in multiplicative depth, requiring relinearization and rescaling to reduce size and noise growth.

3.2. Threshold decryption and secret sharing

This study adopts a threshold decryption mechanism, in which the secret key is never disclosed in plaintext to any single party. The plaintext can only be recovered when at least t participants jointly collaborate. The Shamir secret sharing scheme selects a random polynomial over a finite field F_p, and distributes shares $s_i = f(i)$ to each participant i. Any t shares can reconstruct the secret using Lagrange interpolation.

$$f(x) = s + \sum_{i=1}^{t-1} a_i x^i \quad (4)$$

In this context, the BFV secret key $s \in R_t$ is secret-shared using Shamir's Secret Sharing, either coefficient-wise or over the modulus q, producing shares $\{S_t\}$. Each participant who receives the ciphertext $c = (c_0, c_1)$ computes a partial decryption as follows:

$$v_i = c_1 \cdot s_i \in R_q \quad (5)$$

Then, v_i is sent back to the designated reconstructor. Upon collecting any t valid shares, the reconstructor uses the Lagrange coefficients $\{\lambda_i\}$ to compute:

$$v = \sum_{i=1}^{t} \lambda_i \cdot v_i \, (mod \, q) \quad (6)$$

Finally, the reconstructor locally computes the plaintext:

$$\widetilde{m} = Round \left(\frac{t}{q} (c_0 + v) \right) mod \, t \quad (7)$$

3.3. Zero-knowledge proofs

Zero-knowledge proofs are a class of cryptographic protocols that allow a prover to convince a verifier of the truth of a given statement without revealing any additional information beyond the fact that the statement is true. Formally, let the relation $R \subseteq \{0,1\}^* \times \{0,1\}^*$, and define the language $L_R = \{x | \exists w \, (x, w) \in R\}$. Given a public

statement x and a private witness w, a zero-knowledge proof system consists of an interactive protocol between two parties *(P,V)*, the prover and the verifier, satisfying the following properties:

(1) Completeness: If $(x,w) \in R$, then an honest prover P can convince an honest verifier V to accept with high probability;

(2) Soundness: If $x \notin L_R$, then no (even malicious) prover can convince V to accept, except with negligible probability;

(3) Zero-knowledge: There exists an efficient simulator that can generate a simulated transcript which is computationally indistinguishable from a real interaction between P and V. This ensures that during the proof process, the verifier learns nothing beyond the truth of the statement being proven.

Zero-knowledge proofs (ZKPs) can be categorized into interactive and non-interactive forms, based on the interaction pattern and system model. Interactive ZKPs require multiple rounds of challenge-response exchanges between the prover and verifier to complete the proof process. Non-interactive ZKPs (NIZKs), on the other hand, typically leverage the Fiat–Shamir transform to collapse interaction into a single round, allowing the prover to generate a one-shot, publicly verifiable proof. This makes NIZKs particularly suitable for blockchain and other large-scale, low-interaction environments.

3.4. Blockchain Technology

Blockchain is a decentralized ledger technology built on cryptography and distributed systems. At its core, it organizes transaction data into blocks in chronological order and links these blocks sequentially using hash pointers, forming an immutable chain structure. Each block typically consists of two parts: the block header and the block body, block architecture with Merkle Tree as illustrated in **Figure 1**.

The block header contains metadata such as the version number, hash of the previous block, timestamp, nonce, and difficulty target, which together define the block's identity and consensus constraints. The block body stores all transaction records within the block, and a unique Merkle root is generated via multi-level hashing of the transaction data. When a new block is created, it includes the hash of the previous block in the "previous block hash" field of its header, thereby forming a chained structure. If any transaction data within a block is tampered with, the corresponding Merkle root and all subsequent block hashes would change, making the forgery immediately detectable. This mechanism ensures the traceability, integrity, and tamper-resistance of the entire blockchain.

Depending on the level of trust among participants and access control requirements, blockchain systems are typically categorized into public chains, private chains, and consortium (permissioned) chains: Public blockchains are fully open, allowing any node to join or leave freely. They are well-suited to low-trust, large-scale decentralized environments. Private blockchains are maintained by a single organization and are primarily used for internal governance and data management. Consortium blockchains lie between the two extremes, designed for multi-party collaboration. With controlled identities and permissioned access, they achieve an effective balance between performance and privacy protection.

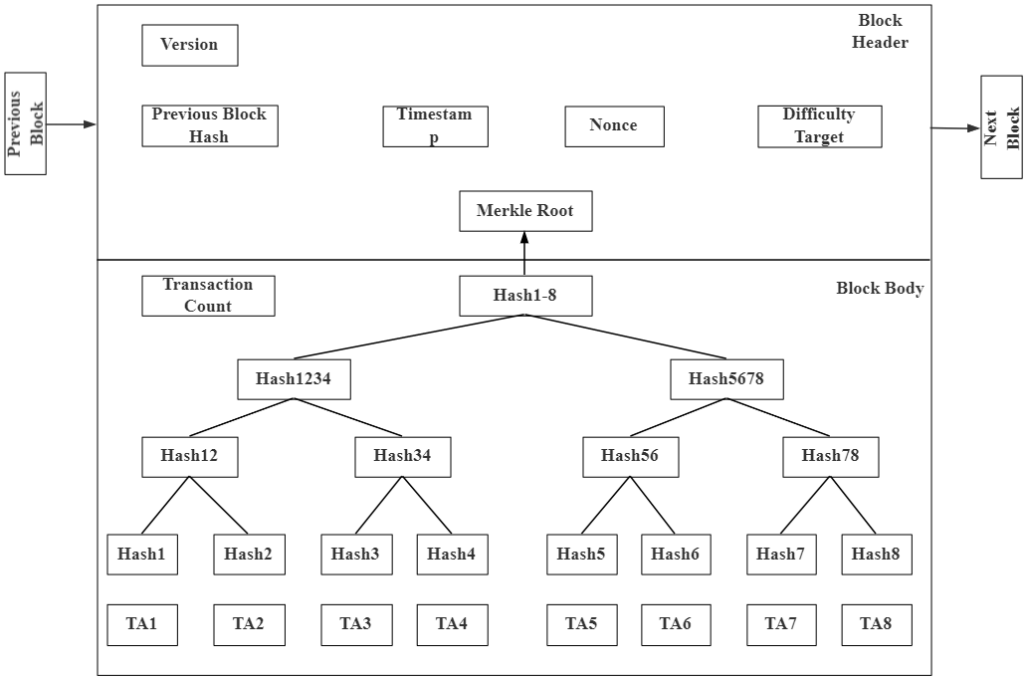

Figure 1. Block architecture with Merkle Tree.

4. Scheme design

This section presents the detailed design of a cross-departmental financial computation framework that integrates homomorphic encryption, zero-knowledge proofs, and blockchain technology. The proposed scheme utilizes the BFV homomorphic encryption algorithm implemented via Microsoft SEAL, ensuring secure computations on encrypted data. Zero-knowledge proofs are employed to guarantee the correctness of computation without revealing sensitive information. At the same time, blockchain and IPFS are leveraged to ensure data privacy, verifiability, and scalability. By adopting a distributed computing framework, the scheme enables secure and efficient collaboration across departments while maintaining strong protection for sensitive data.

4.1. Initialization phase: System parameter generation

During the system initialization phase, the encryption parameters required for homomorphic encryption are first generated. These parameters include the polynomial modulus, coefficient modulus, and plaintext modulus, which together support data processing and computation throughout the encryption process. Key generation is handled by a dedicated key management service. An authorized organization randomly selects:

$$N = p \times q, M = p' \times q' \tag{8}$$

where *p, q, p', q'* are large primes satisfying *gcd(p,q-1)=1*

The parameters are configured as follows:

(1) Polynomial modulus: Set the value of *n* to ensure that the BFV encryption scheme can efficiently handle large-scale data;

(2) Coefficient modulus: Set to BFVDefault (n), which defines the standard coefficient modulus in the BFV

scheme. This determines both the precision and security level of the encryption process;

(3) Plaintext Modulus: Set to Batching (n,26), which ensures sufficient computational precision during encryption to avoid overflow issues when processing plaintext data.

4.2. Key generation and management

The system generates a private key *sk* and a public key *pk*, which are used during the encryption and decryption processes. The private key is securely stored and managed by a key management service (KMS) to ensure its confidentiality. The public key is published and made available to other departments for encryption operations. The keys are generated as follows:

$$sk = \left(\sum_{i=0}^{L-1} a_i \cdot r^i\right) mod\ N, pk = g^{sk}\ mod\ N \tag{9}$$

where a_i are randomly generated coefficients, r is a random factor used in the key generation process, L is the length of the encryption parameter.

4.3. Sales data computation

Within the cross-departmental computation framework, the primary task of the sales department is to encrypt sales data and compute the total sales amount. The process involves three main steps: encrypting the sales data, performing batch summation over the encrypted data, and transmitting the encrypted result to the finance department.

4.3.1. Sales data encryption

The sales department uses a BatchEncoder to encode the sales data and then employs an Encryptor to encrypt the data using the public key. The resulting ciphertext remains encrypted throughout the computation process, ensuring data privacy:

$$C_{sales} = Encrypt\ (pk, pt_{sales}) \tag{10}$$

4.3.2. Batch summation

The encrypted sales data is processed using the sum_all_slots Algorithm 1, which performs a batch summation across all ciphertext slots. This operation calculates the total sales amount across all sales personnel. Internally, it uses rotation and addition operations to ensure that the value in each slot reflects the overall sum of sales amounts. The final ciphertext representing the total sales amount is:

$$C_{total\_sales} = sum\_all\_slots\ (C_{sales}, encoder, gk, evaluator) \tag{11}$$

This ciphertext is then securely transmitted to the finance department.

Algorithm 1: sum_all_slots

1. Initialize tmp;
2. evaluator.rotate_columns(ct, gk, tmp);
3. evaluator.add_inplace(ct, tmp);
4. row_size = encoder.slot_count() / 2;
5. For step = 1 to row_size:

6. evaluator.rotate_rows(ct, step, gk, tmp);
7. evaluator.add_inplace(ct, tmp);
8. Return ct;

4.4. Financial data computation

The primary task of the finance department is to encrypt cost data, compute the total cost, and then perform computations with the total sales amount to derive the total profit. The department first uses a BatchEncoder to encode the cost data and then encrypts it using the Encryptor with the public key *pk*, ensuring the privacy of the cost information:

$$C_{costs} = Encrypt\ (pk, pt_{costs}) \tag{12}$$

The encrypted cost data is then processed using a batch summation, similar to the handling of sales data. The sum_all_slots algorithm is applied to sum all slots in the ciphertext and compute the total cost:

$$C_{total_{cost}} = sum\_all\_slots\ (C_{costs}, encoder, gk, evaluator\ ) \tag{13}$$

Next, the finance department uses the Evaluator to perform subtraction between the encrypted total sales and total cost, yielding the encrypted total profit:

$$C_{profit} = C_{total\_sales} - C_{total\_cost} \tag{14}$$

4.5. Tax department computation

The tax department is responsible for computing taxes based on the total profit and applicable tax rate. In this phase, the department performs a multiplication operation between the encrypted total profit and the encrypted tax rate, and then returns the computed ciphertext to the relevant departments. The tax department first encodes the tax rate and encrypts it using the Encryptor with the public key *pk*:

$$C_{tax\_rate} = Excrypt\ (pk, pt_{tax}) \tag{15}$$

Then, the encrypted tax amount is computed by multiplying the ciphertexts of the total profit and tax rate:

$$C_{tax} = C_{profit} \times C_{tax\_rate} \tag{16}$$

4.6. Integration of zero-knowledge proof and on-chain verification

To ensure the correctness of computation within the cross-department homomorphic computation framework, without revealing any raw data, this scheme integrates ZKPs with blockchain technology. Using the example of verifying the correctness of total sales summation, the scheme demonstrates how consistency between profit and cost, and between tax and profit, can be similarly proven under the same framework. By deploying zero-knowledge proof verification within smart contracts, the solution not only preserves data privacy, but also ensures the verifiability and transparency of the computation results.

The verification target is the correctness of a homomorphically computed batched summation. The sales department seeks to prove that the published total-sales ciphertext was derived from a committed ciphertext vector using a fixed “rotation + binary-tree summation” computation circuit, while also proving that all underlying plaintext values fall within a predefined legal range.

4.6.1. Public inputs

The proof exposes only the following public information:

(1) paramsHash: Hash of the homomorphic encryption public key and encryption parameters, ensuring parameter consistency;

(2) ctVecHash: Hash of the encrypted sales data vector, ensuring consistency of data origin;

(3) ctSumHash: Hash of the total sales ciphertext, ensuring consistency of the computation result;

(4) n, t, row_size: Batching parameters used in the encryption process;

rangeBound: Legal range of sales amount values.

4.6.2. Witnesses

The prover supplies, privately as follows:

(1) s[ ]: Plaintext sales amounts in each slot;

(2) r_enc: Randomness used during encryption;

(3) Intermediate commitments and randomness from the rotation and summation steps in C_SalesSum.

4.6.3. Statement to be Proven

The ZKP system proves the following properties without revealing any plaintext:

(1) Decode(ctVec) = s[ ];

(2) Applying a fixed sequence of row-swaps and intra-row binary tree summation to ctVec yields a ciphertext equal to ctSum;

(3) ∀i, 0 ≤ s[i] ≤ rangebound.

4.6.4. Prover Actions

The prover first encrypts the sales vector using the public key to obtain , uploads it to IPFS, and obtains its content identifier. The total-sales ciphertext is then computed using the fixed aggregation algorithm and likewise uploaded. All public inputs are assembled as:

$$x = \{paramsHash, ctVecHash, ctSumHash, n, t, row_{size}, rangeBound\}$$

A zero-knowledge proof is generated:

$$\pi = ZK \cdot Prove\ (C_{SalesSum}, x; W) \tag{17}$$

where W is the witness, including intermediate commitments and randomness used in the computation.

4.6.5. Smart Contract Verification

The smart contract calls to validate the proof against the declared public inputs. Successful verification certifies that the total-sales ciphertext was correctly derived and that no tampering occurred during computation.

$$Verifier \cdot verify\ (\pi, x) \tag{18}$$

4.6.6. On-Chain and Off-Chain Storage

All zero-knowledge proof data, sales ciphertexts, and computation results were stored off-chain via IPFS. Only hashes, proof metadata, and verification outcomes were recorded on-chain, ensuring lightweight and efficient

blockchain storage.

5. Experimental analysis

In this section, we aim to evaluate the performance of each computation phase involved in cross-departmental financial processing using the BFV homomorphic encryption scheme. By measuring the computation time at different stages on a dataset of 5,000 records, we assess the practical performance and identify potential bottlenecks of homomorphic encryption in real-world applications. The experiments were conducted on a machine equipped with an Intel i7-9700K processor, 16 GB of RAM, and SSD storage, running Ubuntu 20.04 LTS. For the software environment, we used Microsoft SEAL 3.6.5 as the homomorphic encryption library, g++ 9.3.0 as the compiler, CMake 3.16.3 for build configuration, and compiled all code under the C++17 standard. The experiments measured the percentage of execution time spent in each of the following operations within one encryption cycle: SaleEncrypt, TotalSales, CostEncrypt, TotalCost, ProfitTime, Taxtime. The breakdown of execution time across these operations is illustrated in **Figure 2**.

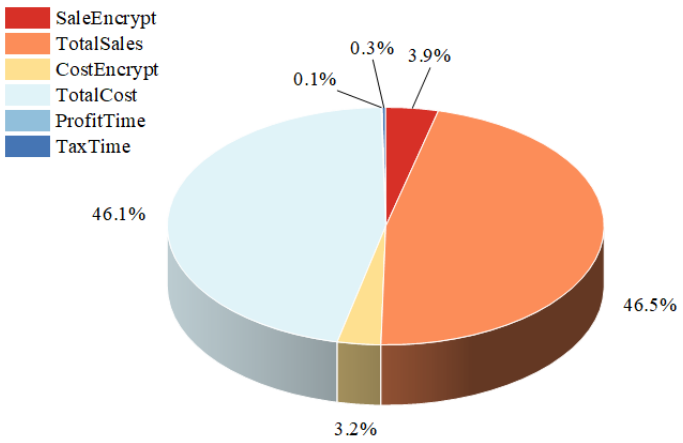

Figure 2. Percentage of time in each phase.

6. Conclusion

This paper addresses the challenges of privacy leakage and compliance auditing in cross-departmental data collaboration by proposing a secure computation framework that integrates homomorphic encryption, threshold decryption, and blockchain technology. By designing a unified parameter system and a multi-phase encrypted computation workflow, the framework enables joint analysis of sales, cost, and tax information without exposing raw data. It effectively ensures data confidentiality and verifiability during interdepartmental data exchange. Experimental results demonstrate that the proposed scheme achieves a good balance between computational performance and security, validating its feasibility and efficiency in multi-party trusted collaboration scenarios. Future research will examine lightweight zero-knowledge proof mechanisms to further reduce verification complexity and communication overhead, as well as the integration of multi-chain interoperability and privacy auditing frameworks, with the goal of developing a scalable, cross-organizational computation platform that ensures both strong privacy protection and regulatory compliance. This will further support trustworthy data sharing in complex business environments and provide practical solutions for privacy and regulatory requirements.

Disclosure statement

The authors declare no conflict of interest.

References

[1] Acar A, Aksu H, Uluagac A, et al., 2018, A Survey on Homomorphic Encryption Schemes: Theory and Implementation. ACM Computing Surveys, 51(4): 1–35.

[2] Wirth F, Meurers T, Johns M, et al., 2021, Privary-Preserving Data Sharing Infrastructures for Medical Research: Systematization and Comparison. BMC Medical Informatics and Decision Making, 21(1): 242.

[3] Gorantala S, Springer R, Gipson B, 2023, Unlocking the Potential of Fully Homomorphic Encryption. Communications of the ACM, 66(5): 72–81.

[4] Gentry C, 2010, Computing Arbitrary Functions of Encrypted Data. Communications of the ACM, 53(3): 97–105.

[5] Zhang Y, Ma Z, Meng J, 2025, Auditing in the Blockchain: A Literature Review. Frontiers in Blockchain, 8(2025): 1549729.

[6] Čuřík P, Roderik P, Zajac P, 2022, Practical Use of Secret Sharing for Enhancing Privacy in Clouds. Electronics, 11(17): 2758.

[7] Tawfik A, Ahwal A, Eldien A, et al., 2025, PriCollabAnalysis: Privacy-Preserving Healthcare Collaborative Analysis on Blockchain using Homomorphic Encryption and Secure Multiparty Computation. Cluster Computing, 28(3): 191.

[8] Li G, He D, Guo B, et al., 2020, Blockchain Privacy Protection Algorithms based on Zero-Knowledge Proof. Journal of Huazhong University of Science and Technology (Natural Science Edition), 48(7): 112–116.

[9] Fujisaki E, Okamato T, 1997, Statistical Zero Knowledge Protocols to Prove Modular Polynomial Relations. Annual International Cryptology Conference.

[10] Mohammed M, Wahab H, 2024, Enhancing IoT Data Security with Lightweight Blockchain and Okamoto Uchiyama Homomorphic Encryption. Computer Modeling in Engineering & Sciences, 138(2): 1731–1748.

[11] Chen R, Li Y, Zhang Y, et al., 2025, Bitcoin-Compatible Privacy-Preserving Multi-Party Payment Channels Supporting Variable Amounts. IEEE Transactions on Information Forensics and Security, 2025(20): 2984–2998.

[12] Koundinya K, Gautham S, 2021, Two-Layer Encryption based on Paillier and Elgamal Cryptosystem for Privacy Violation. International Journal of Wireless and Microwave Technologies, 11(3): 9–15.

[13] Adarbah H, Kiraz M, Kardas S, et al., 2024, A New Framework for Enhancing VANETs through Layer 2 DLT Architectures with Multiparty Threshold Key Management and PETs. Future Internet, 16(9): 328.

[14] Zheng B, 2018, Scalable and Privary-Preserving Data Sharing based on Blockchain. Journal of Computer Science and Technology, 33(3): 557–567.

[15] Si H, Li W, Su N, et al., 2024, A Cross-Chain Access Control Mechanism based on Blockchain and the Threshold Paillier Cryptosystem. Computer Communications, 223(2024): 68–80.

[16] Zhu L, 2020, Data Security and Privacy in Bitcoin System: A Survey of Computer Science and Technology, 35(4): 843–862.