GDPR-Compliant e-KYC using Redactable Blockchain and Zero Knowledge Proof

Piphattra Sreekongpan, Supawit Nakprame, Surabordin Sungchai
School of ICT, Sirindhorn International Institute of Technology, Thammasat University, Pathum Thani, Thailand
sr_pipattra@outlook.com, 6422770287@g.siit.tu.ac.th, 6422782514@.siit.tu.ac.th

***Abstract*—** The increasing reliance on electronic Know Your Customer (e-KYC) systems by financial institutions necessitates compliance with stringent privacy regulations, such as the General Data Protection Regulation (GDPR), to safeguard personal data. However, traditional blockchain-based solutions struggle to accommodate privacy regulations like GDPR due to their immutability, especially in the context of a user's right to request the deletion of personal data. This paper introduces a blockchain-based framework that integrates redactable and privacy-aware techniques, such as chameleon hashing and zero-knowledge proofs (ZKPs), to provide a GDPR-compliant e-KYC solution that balances accountability, transparency, and user privacy. The framework not only enhances accountability and privacy but also addresses the scalability and performance issues typically associated with traditional blockchain-based solutions. The proposed framework demonstrates improved performance metrics compared to conventional blockchains, highlighting the potential for practical implementation in real-world e-KYC and data privacy applications.

Keywords—eKYC, GDPR, redactable blockchain, zero knowledge proofs

I. INTRODUCTION

The rapid growth of e-commerce has heightened the demand for secure, reliable, and efficient customer verification mechanisms. Among these, KYC process has emerged as a widely adopted approach to verify customer identities, preventing fraud, money laundering, and other financial crimes. By collecting and verifying personal information, such as government-issued IDs and proof of address, businesses can confirm the legitimacy of customers, thereby mitigating risks associated with online transactions. As digital platforms continue to expand, KYC has become an essential component in maintaining trust, security, and regulatory compliance across the e-commerce industry.

However, traditional KYC processes often rely on manual, paper-based verification systems that are not only time-consuming and error-prone but also susceptible to data breaches. The advent of blockchain technology has introduced a paradigm shift, enabling decentralized and secure solutions for managing sensitive customer information. Despite its advantages, privacy concerns regarding the storage and sharing of personal data remain a pressing challenge.

e-KYC has gained widespread adoption among financial institutions (FIs) and banks worldwide, facilitating backend customer authentication without the direct exchange of credentials between parties. Modern e-KYC systems employ cryptographic-based access control solutions and advanced authentication mechanisms, such as two-factor authentication (2FA) and multi-factor authentication (MFA). Additionally, secure data exchanges between FIs are achieved through encryption and adherence to strict authorization policies. Blockchain technology further enhances e-KYC by improving security, traceability, and efficiency in banking applications. Its decentralized and immutable nature offers a secure, cost-effective way to streamline KYC processes, ensuring customer privacy, reducing fraud, and complying with regulatory requirements. This makes blockchain an ideal solution for digital identity verification across the e-commerce and financial sectors, reducing dependence on centralized trusted third parties.

Despite these advancements, existing e-KYC systems face several unresolved challenges. First, scalable and anonymous authentication is critical, particularly when processing e-KYC requests involving a large number of customers and non-host FIs. Current e-KYC systems rely heavily on centralized authentication mechanisms like 2FA and MFA, managed by the core systems of individual FIs. These centralized approaches incur high communication costs and create bottlenecks during peak request volumes, especially in scenarios requiring large-scale e-KYC validations. For example, when multiple non-host FIs simultaneously request e-KYC validation for loan applications, centralized authentication systems often face delays due to network congestion and computation overload on the host FI.

Second, secure and dynamic data portability is a pressing requirement in real-world applications. Financial institutions frequently need to share customer data securely and efficiently during processes like loan approvals, financial audits, or cross-border transactions. However, existing e-KYC systems struggle to provide flexible and secure mechanisms for dynamic

authentication and data transfer between multiple FIs while ensuring compliance with regulatory frameworks.

Third, compliance with privacy regulations such as GDPR [1] poses a significant challenge. Blockchain's immutable nature conflicts with the "right to be forgotten" clause, making it difficult to erase or redact personal data upon customer request. Additionally, ensuring that e-KYC verification requests remain confidential and occur only with explicit customer consent is a persistent issue in blockchain-based e-KYC systems.

To address these challenges, we propose a novel secure and flexible, GDPR-compliant e-KYC scheme leveraging a redactable blockchain architecture. Our solution integrates advanced techniques to ensure secure, scalable, and efficient customer verification. First, we introduce an optimized authentication mechanism that combines Zero-Knowledge Rollup (ZK-Rollup) and a novel sharded proof verification strategy. This mechanism enables transparent validation of customer credentials while efficiently handling large volumes of e-KYC requests. e-KYC transactions are grouped based on proximity to distributed verification services, batched into a single succinct Zero-Knowledge Proof off-chain, and then submitted to the blockchain. This approach significantly reduces on-chain computation and storage requirements, improving throughput and scalability.

Second, to facilitate secure and dynamic data portability, our scheme incorporates re-encryption. This mechanism enables encrypted data transfer between host and requesting FIs without exposing plaintext information. The system dynamically adjusts to varying authentication requirements and ensures secure access to customer data for critical operations, such as loan approvals and regulatory audits.

Third, to ensure compliance with privacy regulations, we propose an optimized redactable blockchain protocol that leverages the Chameleon Hash algorithm for controlled data redaction. Our approach integrates off-chain redaction execution to improve scalability and efficiency, while the security of the redaction process is ensured through our proposed dynamic key rotation mechanism. This design guarantees privacy compliance, data integrity, and adaptability in managing sensitive customer information. To the best of our knowledge, we provide the first attempt in addressing the "right to be forgotten" issue of GDPR in blockchain-based e-KYC.

In summary, the key contributions of our proposed system include:

1. **Scalable and Efficient Authentication**: The integration of ZK-Rollup and sharded proof verification facilitates the handling of large-scale e-KYC requests with minimal on-chain computational overhead, ensuring high throughput and efficiency. Off-chain proof generation and batching significantly reduce computational and storage costs, making the system more economical and scalable for real-world applications. +Sharded BC
2. **Dynamic and Secure Data Portability**: The system incorporates adaptive proxy re-encryption to enable secure and flexible data sharing between host and requesting financial institutions (FIs). This ensures that critical operations like loan approvals and regulatory audits are supported seamlessly.
3. **Secure and Optimized Redaction process**: By leveraging the Chameleon Hash algorithm, combined with off-chain redaction execution, the protocol allows controlled modifications to stored data without compromising blockchain integrity. Additionally, the implementation of dynamic key rotation enhances the security of the redaction process and ensures long-term compliance with privacy regulations, including GDPR.

II. RELATED WORK

This section discusses works entailing e-KYC systems and literature addressing the redactable blockchain.

e-KYC System

Kumar et al. [8] proposed a blockchain-based approach to decentralize the storage of personal data in the KYC process, addressing inefficiencies and operational redundancies in traditional methods. Their scheme focuses on data sovereignty by encrypting users' Personally Identifiable Information (PII), which is stored in a QR code format. Additionally, the system emphasizes enhanced user privacy and cost reduction through decentralized coordination among banks, regulators, and stakeholders.

Mamun et al. [9] proposed a secure and transparent KYC document verification system for banking using IPFS and blockchain technology. Their approach streamlines the KYC process by enabling customers to complete the process once at a single bank, generating a hash value via the IPFS network and sharing it securely through blockchain.

Fugkeaw [2] introduced e blockchain-based e-KYC scheme, known as e-KYC TrustBlock, which leverages ciphertext-policy attribute-based encryption (CP-ABE) combined with client consent enforcement to ensure trust, security, and compliance with privacy standards. Furthermore, the scheme incorporates attribute-based encryption to facilitate privacy-preserving and fine-grained access control for sensitive transactions stored on the blockchain. A smart contract is also developed to create and enforce digital consents signed by customers, with these consents systematically recorded on the blockchain. The author also presented a policy update algorithm to enable efficient re-encryption, utilizing a simplified policy tree structure.

Suga [3] introduced the e-KYC-e model, leveraging the ERC735 claim-style credentials in the Ethereum blockchain to enhance digital identity verification. The model utilizes decentralized identifiers (DIDs) and self-sovereign identity

(SSI) concepts, enabling individuals to control their digital identities. Through a metaphor of "cotton candy," the paper illustrates how multiple attributes (claims) can be linked to a specific identifier, creating a comprehensive and verifiable digital identity. The proposed system demonstrates its applicability by providing digital proof of official certificates, such as enrollment records, within a blockchain framework.

Patil and Sangeetha [7] proposed a blockchain-based decentralized KYC verification framework for banks, leveraging the Ethereum blockchain to enhance the efficiency, security, and transparency of the KYC process. The framework enables all banks within the blockchain network to verify the legitimacy of customer-provided data through a voting mechanism. The KYC status of a customer is stored on the blockchain based on the consensus achieved through this process. A unique aspect of the framework is its provision for banks to vote on the behavior of other banks, allowing the removal of entities that tamper with KYC data.

Schlatt et al. [11] proposed a framework for digital KYC processes leveraging blockchain-based SSI to address inefficiencies and privacy concerns in traditional KYC methods. The framework employs a design science research approach to integrate SSI, enabling customers to maintain control over their personal data while ensuring compliance with data protection regulations. The study derives design principles that theorize blockchain's role in facilitating SSI, offering a privacy-preserving and efficient solution to KYC challenges.

Takaragi et al. [13] proposed a privacy-preserving eKYC framework tailored for Central Bank Digital Currencies (CBDCs), integrating delegatable anonymous credentials (DAC) and zero-knowledge range proofs (ZKRP). By leveraging zero-knowledge proofs, the system ensures that sensitive information, such as time stamps and ID registration validity, can be verified without revealing their content. This approach enables privacy-enhanced public key infrastructure (PKI) and supports self-sovereign identity management, contributing to a sustainable financial system.

Jaber et al. [4] proposed a conceptual framework for integrating blockchain technology into the banking sector's e-KYC systems. The framework emphasizes enhancing security, operational efficiency, and real-time verification through features such as digital documentation, biometric verification, electronic consent, and blockchain integration. The study highlights the potential of blockchain to streamline lending processes, reduce counterparty risk, and improve return on investment while addressing challenges like regulatory compliance, data privacy, scalability, and interoperability.

Bhatia et al. [10] proposed a cost-efficient blockchain-based e-KYC platform that leverages biometric verification to address the limitations of traditional and digitized KYC processes. The platform allows users to complete video-based KYC verification after making an ether payment, upon which they receive a unique KYC key. This key enables FIs to verify user authentication directly on the blockchain without requiring repeated submission of documents, ensuring privacy and reducing redundancy. By leveraging blockchain's decentralized, transparent, and immutable properties, the proposed platform enhances security, minimizes operational costs, and streamlines the KYC process.

Patkar et al. [12] proposed a privacy-preserving and trustworthy e-KYC system utilizing blockchain technology to address the inefficiencies and privacy concerns associated with traditional centralized KYC processes. The system leverages a decentralized blockchain network to securely store and manage customer identity data, eliminating the need for repetitive verification across multiple financial institutions. By employing cryptographic techniques, the proposed framework enhances data security, ensures transparency in data sharing with customer consent, and reduces operational costs. This approach optimizes the KYC verification process, fostering trust between customers and financial institutions.

Recently, Ahmed et al. [6] introduced a novel approach to enhance the security and efficiency of e-KYC systems by integrating blockchain technology, Web 3.0, and quantum computing. The proposed framework addresses limitations in current e-KYC solutions, such as high key management costs and reliance on traditional encryption, by employing quantum encryption key methodologies to ensure stronger data protection and confidentiality.

Nevertheless, all the works mentioned above highlight significant advancements in e-KYC systems; however, they do not fully address the inherent conflict between blockchain's immutability and regulatory requirements such as the GDPR. This conflict poses challenges in managing erroneous or sensitive data.

Redactable Blockchain

Recent advancements in redactable blockchain models offer promising solutions to balance data integrity with compliance, privacy, and usability. These solutions address the inherent immutability of blockchain systems while enabling selective data modification to meet regulatory requirements such as GDPR. Various techniques and frameworks have been proposed, leveraging mechanisms like chameleon hash functions, trapdoor hash functions, privacy-preserving methods, and integrations with revocable IPFS to enhance redactable blockchain systems.

One widely adopted technique for enabling redaction in blockchains is the use of chameleon hash functions. These functions allow authorized users to modify blockchain data securely without compromising its cryptographic integrity. Dong et al. [18] employed a chameleon hash function in conjunction with multi-authority attribute-based encryption to enable authorized individuals to redact or replace content securely. This approach ensures that data visibility is restricted to authorized users, addressing privacy concerns in consortium blockchain settings. Zhang et al. [19] extended this technique by integrating dynamic proactive secret sharing (DPSS), chameleon hash, and digital signatures into their trust-based redactable blockchain. Their model ensures traceability before and after modifications and securely restricts modification privileges. Similarly, Wang et al. [20] used decentralized

chameleon hash functions to support rapid and efficient copyright maintenance, allowing the modification of infringing content in educational data. Guo et al. [17] developed an attribute-based chameleon hash function to facilitate transaction-level redactions of Electronic Health Records (EHR), ensuring compliance with privacy regulations while enabling secure and efficient redactions.

Several studies have focused on ensuring GDPR compliance through redactable blockchain models that support selective record removal and flexible access control. Yeh et al. [14] proposed a GDPR-compliant mechanism combining a redactable blockchain with revocable IPFS. Their approach simplifies access control through enhanced proxy re-encryption, eliminating the complexities of group key management while allowing selective removal of records. Xu et al. [21] introduced a privacy-preserving healthcare blockchain system (PRHBS) that integrates trapdoor-based chameleon hash functions, attribute-based encryption, and puncturable encryption. This system enables fine-grained block-level data redaction and secure data sharing, aligning with GDPR requirements while ensuring data confidentiality in multi-user environments.

Another key innovation in redactable blockchain systems is the use of trapdoor hash functions to enable fine-grained control over modifications. Zhou et al. [15] utilized a trapdoor hash-based approach to allow efficient transaction editing, deletion, and smart contract repair. Their solution provides low-overhead and flexible permission control, demonstrating significant potential for real-world applications. Similarly, Xu et al. [21] incorporated trapdoor hash functions in their PRHBS system to facilitate secure and flexible redactions in the healthcare sector.

In addition to supporting data redaction, several models prioritize privacy-preserving mechanisms to protect user anonymity. Heo et al. [16] introduced zk-SNARKs to enhance privacy by generating one-time cryptographic keys for transactions, ensuring that user identities cannot be tracked. Their model further obscures cryptographic keys and signatures, providing an additional layer of security. To optimize performance, Heo et al. proposed a redaction fee scheme that incentivizes users to remove data rather than modify it, ensuring efficient ledger management and minimal performance overhead.

The integration of revocable IPFS with redactable blockchain models is another noteworthy approach to improving data management and ensuring compliance. Yeh et al. [14] leveraged revocable IPFS to enable selective removal of files, maintaining both privacy and usability in their GDPR-compliant framework. Guo et al. [17] employed a similar method for managing EHRs, enabling rapid redactions and ensuring compliance with privacy regulations.

Despite the advancements in redactable blockchain models highlighted above, key challenges remain. Particularly, the scaling redaction processes for high-throughput systems, enabling dynamic data portability across institutions, and minimizing the computational and storage overhead associated with frequent redactions are not fully addressed.

III. PRELIMINARIES

This section provides an overview of the foundational technologies and concepts utilized in our framework, which combines e-KYC, GDPR, blockchain, chameleon hashing, and zero-knowledge proofs. We also discuss related works, highlighting the existing literature in these fields.

ZERO-KNOWLEDGE PROOF

A Zero-Knowledge Proof must satisfy three fundamental properties: Completeness, which ensures that if the statement is true, an honest verifier will be convinced by an honest prover with a high probability of correctness, thereby guaranteeing reliability in verification processes; Soundness, which guarantees that if the statement is false, no dishonest prover can convince an honest verifier with non-negligible probability, ensuring robustness against fraudulent claims and adversarial manipulations; and Zero-Knowledge, which means that if the statement is true, the verifier learns nothing beyond the validity of the statement, thereby maintaining strict privacy and security of the underlying information. These three principles collectively form the backbone of ZKP protocols, enabling secure and private verification. Mathematically, a ZKP protocol consists of a tuple (P, V, S) where P is the prover, who attempts to convince the verifier; V is the verifier, responsible for validating the proof; and S is a simulator that can reproduce the verifier's output without access to the witness, ensuring that no extraneous information is leaked in the process. This formalism provides a strong cryptographic framework that underpins numerous security-sensitive applications.

CHAMELEON HASHING

The system utilizes Chameleon Hashing to enable selective modification (redaction) of blockchain data without disrupting its structure or cryptographic integrity, making the blockchain GDPR-compliant, and fulfilling the requirement of the "right to be forgotten."

1. **HGen** $(1^k) \rightarrow (hk, tdk)$ **:** The algorithm generates a trapdoor key (tdk) that enables modifications and a hash key (hk) from a prime number p, its subgroup, and a generator of quadratic residues modulo g.
2. **Hashing:** To hash a message m, the algorithm combines it with a randomly generated value r and applies a collision-resistant hash function H.

$$h := r - (y \cdot H(m||r) + g^s) \bmod p$$

This hash value h uniquely identifies m while being tied to randomness r and the public hash key hk.

3. **Collision:** If a message m needs to be updated to m', the trapdoor key (tdk) is used to adjust r so that the new hash value remains unchanged, thus preserving the Blockchain integrity.

$$r' = h + g^{k} \bmod p \text{ where } k \text{ is a new random scalar}$$

$$s' = k - H(m'||r') \cdot x \bmod p \text{ where } x = tdk$$

Obtaining new values r' and s' that the message m' remains the same hash value as the original m.

$$h = Hash(hk, m, r, s) = Hash(hk, m', r', s')$$

To optimize performance and reduce computational costs, the redaction process occurs off-chain, potentially on a local computer. This approach minimizes the burden on blockchain nodes, as only the updated hash values and proof of redaction are recorded on-chain. The actual computation for adjusting r' and s' is performed externally, ensuring that blockchain integrity is maintained without requiring expensive on-chain operations.

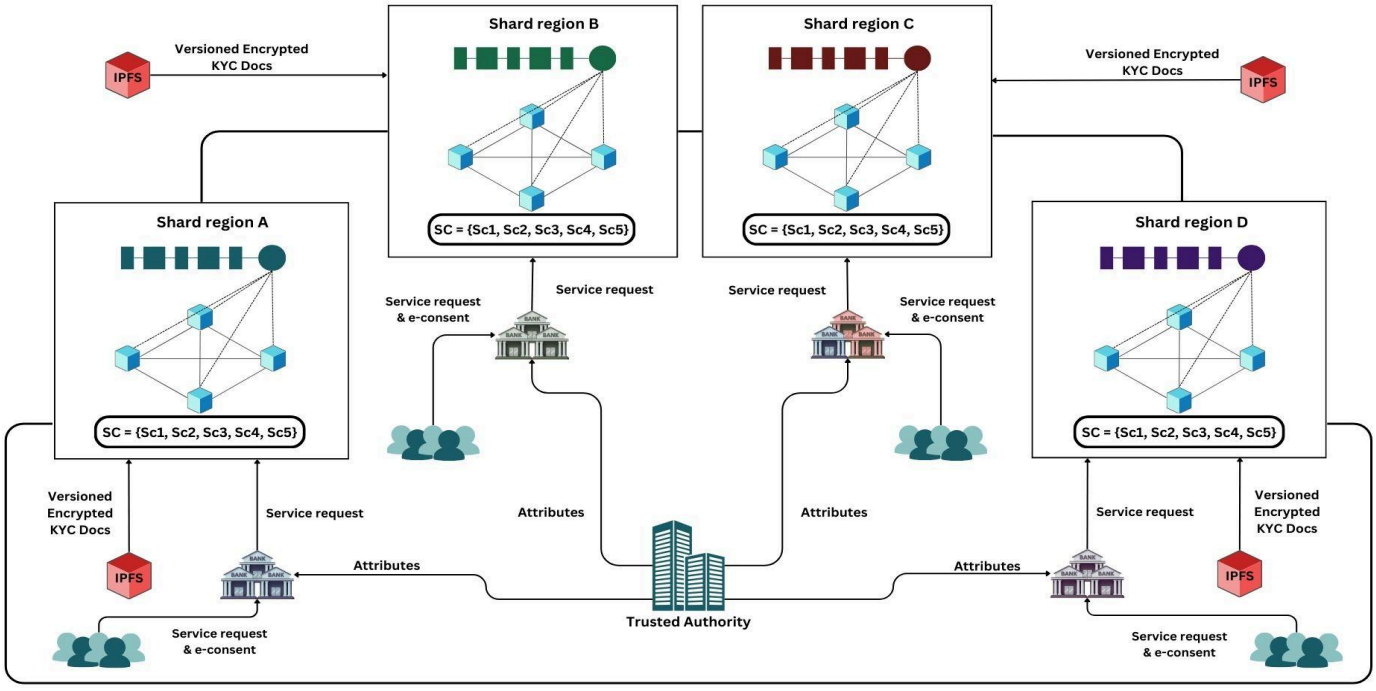
IV. OUR PROPOSED SCHEME

A. SYSTEM MODEL

We propose a GDPR-compliant e-KYC system using a redactable blockchain. Figure 1 provides an overview of the system model, which consists of the following entities:

Fig. 1. System model

1. Trusted Authority (TA) serves as the central trusted entity responsible for generating the public parameters (PK) and the master secret key (MSK) of the system. While the MSK is kept confidential, the PK is publicly available for use by financial institutions (FIs) and customers. The TA also issues cryptographic secret keys to each FI, ensuring secure access to encrypted customer data. Furthermore, this entity validates specific customer attributes upon requests from FIs, enabling secure and efficient e-KYC operations.
2. Customers are the users of financial institutions, such as Bank A or Bank B, who enroll in the blockchain-based e-KYC system. Each customer possesses a unique cryptographic key pair (**Public/Private keys**) used for encrypting and decrypting their personal credential data. Before interacting with financial institutions, customers must provide **e-consent**, digitally signed using their private key ($PrivKey_{Cust\_ID}$). This consent grants FIs permission to store, process, and utilize the customer's KYC data. To ensure non-repudiation and accountability, the e-consent is stored immutably on the blockchain.



3. Financial Institutions (FIs): Financial institutions, such as Bank A and Bank B, act as the primary service providers interacting with the customers. They facilitate the e-KYC process by managing customer registration, verifying credentials, and ensuring data privacy and security. FIs interact with smart contracts to execute processes like document redaction, data encryption, and KYC verification.
4. IPFS: The InterPlanetary File System (IPFS) serves as the distributed storage layer for the encrypted KYC documents. Once the FI receives the customer's KYC documents, the files are encrypted and uploaded to the IPFS, generating a hash value (h) using SHA-256. This hash serves as a unique index to retrieve the document from the IPFS storage. The Distributed Hash Table (DHT) stores the mapping between the customer's citizen ID and the encrypted document, making it accessible but secure. IPFS ensures that KYC documents are stored in a tamper-resistant manner, enabling easy retrieval when required.
5. Consortium Blockchain is employed to maintain transaction logs of all KYC-related activities, ensuring integrity, privacy, and transparency across multiple financial institutions and stakeholders. The consortium blockchain is governed by a group of trusted FIs, such as Bank A and Bank B. To enhance scalability and decentralization, the blockchain is divided into shards, where each shard corresponds to a specific region or group of FIs. Sharding enables parallel processing of transactions within a shard while maintaining communication across shards for seamless collaboration. This structure ensures efficient governance by trusted participants while maintaining high levels of decentralization.
6. Smart Contracts: The system utilizes a set of smart contracts (SCs) to automate and manage the entire e-KYC lifecycle. These smart contracts perform various roles, as outlined below:
   e-Consent Generation Smart Contract (SC1):
   This smart contract handles the coordination of e-consent generation between customers and FIs. It allows FIs to request consent from customers to process their KYC documents. Once the customer provides digitally signed consent, it is immutably recorded on the blockchain. SC1 also supports **consent revocation**, enabling customers to withdraw their consent at any time.
   customer Enrollment Smart Contract (SC2):
   This contract is responsible for authenticating new users, enrolling them in the e-KYC system, and uploading their encrypted credentials to IPFS. SC2 ensures that only legitimate customers are onboarded securely.
   Blockchain transaction encryption Smart Contract (SC3): This smart contract encrypts the e-KYC transactions before updating onto the blockchain.
   e-KYC Document Verification Smart Contract (SC4): SC3 verifies the authenticity and integrity of the customer's KYC documents stored in IPFS. By comparing the stored hash value with the computed hash of the retrieved document, this contract ensures that the data remains untampered and valid.
   Redaction Management Smart Contract (SC5):
   This smart contract facilitates the redaction process using chameleon hashing. It ensures that modified data retains hash consistency with the blockchain records while accurately reflecting changes to the underlying data. This capability supports compliance with GDPR's "right to be forgotten" without compromising the blockchain's integrity.

B. System Process

This section describes the system process of our proposed e-KYC model. Table 1 presents a list of notations and symbols used in this paper.

Notation	Meaning
M	Personally Identifiable Information of one customer.
$PrivK_{FI_{ID}}$	Private key of the Financial institute.
$PrivDK_{FI_{ID}}$	Private delegate key of the Financial institute.
pk	Proving keys used to enable ZKP during the e-KYC process.
vk	Verifying keys used to enable ZKP during the e-KYC process.
Tdk	Trapdoor key which is used in the redaction phase.
$Cert_{FI_{ID}}$	Certification of each FIs that TA generated after validation.
FI_{source}	A FI who provides customer data to the requested FIs.
FI_{req}	A FI who requested customer data from the source FIs.
tk_{source}	A transformation key of the source FI.
Sig_{source}	A signature of the source FI.

Phase 1: Setup phase

In this phase, the Trusted Authority (TA) initializes the cryptographic variables and deploys the system's smart

contracts. Keys are generated using Elliptic Curve Cryptography (ECC) with the P-256 (NIST curve) standard.

1) The TA generates the Master Secret Key (Msk_{α}) and the corresponding public key (P_{α}). For Master Secret Key (Msk_{α}), the TA selects a random scalar α from the finite field of the elliptic curve $Z_{q'}^{*}$, where q is the prime order of the elliptic curve group $Msk_{\alpha} = \alpha$. Public Key (P_{α}): Using the generator point P of the elliptic curve group, the TA computes the corresponding public key: $P_{\alpha} = \alpha \cdot P$.

2) The TA computes hash parameters to map specific inputs to points on the elliptic curve. These parameters support various cryptographic operations within the system:

- Hash of Metadata of Personally Identifiable Information (PII): $H_{PII} = H_1(metadata)$. This hash value is used to securely reference metadata stored on the blockchain.
- Key Generation for Financial Institutions: The following hash function is used to generate unique cryptographic keys for financial institutions. $H_{FI} = H_2(FI_{ID})$.
- Transformation Key for Redaction: This hash function facilitates secure and efficient data redaction processes. $H_{Redaction} = H_3(Redaction\_Key)$.
- Hashed Index of Files Stored on IPFS: It maps and securely retrieves files stored in IPFS. $H_{Index} = H_4(IPFS_{CID})$.

Phase 2: Key Generation

This phase defines the cryptographic keys used by FIs and data owners. It involves interactions between the TA and FIs to generate secure key pairs and establish relationships.

A. Entity-Side Operations (FI): The Financial institute generates a random value $r_{FI_{ID}}$ for computing pseudo-identifier $pid_{FI_{ID}}$ and compute $R_{FI_{ID}}$ which is then grouped together and sent to the Authority.

The initial inputs are ($params$) and ($id_{FI_{ID}}$), the unique identifier for each financial institution. These inputs are used for the Key Generation Algorithm with the process below:

1) Generate Random Scalar $r_{FI_{ID}} \in Z_q^*$

2) Compute $R_{FI_{ID}} = r_{FI_{ID}} \cdot P$ where P is the elliptic curve generator point.

3) Use a hash function H2 to compute the pseudo-identifier $pid_{FI_{ID}} = H_2(r_{FI_{ID}} \oplus id_{FI_{ID}}) \cdot P \,||\, id_{FI_{ID}}$

4) Finally, FI sends ($R_{FI_{ID}}, pid_{FI_{ID}}$) to the TA.

B. Trusted Authority-Side Operations: The TA once received $R_{FI_{ID}}, pid_{FI_{ID}}$ from the FI then, chooses a random value r_A for certificate generation and validates the data that were sent by the FIs. The certificate generation is returned to FIs so that they can use it to finalize their key pair.

1) TA generate a random scalar $r_A \in Z_q^*$.

2) Compute $R_A = r_A \cdot P$.

3) Derive the certificate by combining both random numbers together $Cert_{FI_{ID}} = R_{FI_{ID}} + R_A$.

4) Use a hash function H2 to compute Auxiliary Scalar r_{aux} so that certificate cannot be forged $r_{aux} = H_2(Cert_{FI_{ID}} || pid_{FI_{ID}}) \cdot r_A + \alpha$.

5) Then, TA returns ($Cert_{FI_{ID}}, r_{aux}$) to the FI.

C. Entity-Side Finalization (FI): The FI uses the received data to compute its private and public keys.

1) Combine its random scalar $r_{FI_{ID}}$ and r_{aux} and the certificate $Cert_{FI_{ID}}$ to compute their Private Key $PrivK_{FI_{ID}} = H_2(Cert_{FI_{ID}} || pid_{FI_{ID}}) \cdot r_{FI_{ID}} + r_{aux}$.

2) Compute $PubK_{FI_{ID}} = PrivK_{FI_{ID}} \cdot P$.

3) Verify the public key to make sure it matches TA's certificate $PubK_{FI_{ID}} = H_2(Cert_{FI_{ID}} || pid_{FI_{ID}}) \cdot Cert_{FI_{ID}} + P_{\alpha}$.

D. Delegate Key Pairs Generation: Delegate key pairs are generated for secure redaction and delegation operations.

- Secret key $PrivDK_{FI_{ID}} \in Z_{q'}^*$.
- Public key $PubDK_{FI_{ID}} = PrivDK_{FI_{ID}} \cdot P$.

E. Symmetric Key Generation

Use the delegate private key and metadata to derive a unique AES symmetric key. This key is stored securely in the FI's IPFS and used for decrypting ciphertext during the e-KYC phase.

$$SymKey = H_2(PrivDK_{FI_{ID}} || NatID)$$

Phase 3: Customer Enrollment phase

This phase involves securely registering customer information in the system and adding it to the blockchain. The FIs are responsible for collecting customer data, obtaining explicit consent, and enrolling the customer using smart contracts. Two separate file types are managed in this phase: customer metadata and encrypted documents. In this phase, there are four steps as follows:

1) Obtain Customer Data

Customers provide their personal information required for registration. This information is grouped as $M = \{CustID, NatID, Name, Addr, DoB\}$

- CustID**:** Unique Customer ID assigned by the FI.
- NatID**:** National Identification Number.
- Name**:** Full name of the customer.
- Addr**:** Customer address.
- DoB**:** Date of Birth.

2) Generate e-consent Before adding customer data to the system, the FI must obtain explicit e-consent from the customer to ensure compliance with GDPR. The **e**-Consent Generation Smart Contract (SC1) is triggered to facilitate this process. This consent can include:

- **Permission**: To store and process data on the blockchain and IPFS.
- **Usage Information**: Details of how the data will be used.
- **Revocation Rights**: Rights to withdraw consent at any time.

Below shows the algorithm executed by e-Consent Generation Smart Contract **(**SC1**)**:

Algorithm 1: e-Consent Generation
Function GenerateEConsent(CustID, NatID, Permission, UsageDetails, FI_ID)

```
// Step 1: Verify Customer and FI details
Verify(CustID, NatID)
Verify(FI_ID)

// Step 2: Record Consent Information
ConsentDetails = {CustID, NatID, Permission, UsageDetails}
CustSig = Sign(ConsentDetails, PrivKey_CustID)
Consent = {ConsentID: Hash(ConsentDetails), CustID, NatID, CustSig}

// Step 3: Store Consent on Blockchain
RecordOnBlockchain(Consent)

// Step 4: Return Consent for FI Records
Return Consent
End Function
```

3) Customer Enrollment via Smart Contract

Once the customer's data and consent have been collected, the FI triggers the Customer Enrollment Smart Contract (SC2) to securely store customer information. The procedure of SC2 is presented as follows:

Algorithm2: Customer Enrollment
Function EnrollCust(*CustID, NatID, FI_ID, Metadata, Consent*)

```
// Step 1: Verify Consent
VerifyConsent(ConsentID: Consent.ConsentID)
// Step 2: Compute Metadata Hash
H_Metadata = Hash(Metadata)
// Step 3: Encrypt Customer Data and Upload to IPFS
SymKey = GenerateSymKey(PrivDK_FI_ID, NatID)
EncryptedData = AES_Encrypt(Metadata, SymKey)
CID = UploadToIPFS(EncryptedData)
// Step 4: Record Enrollment on Blockchain
EnrollmentTX = {CustID, NatID, H_Metadata, CID, FI_ID, Consent.ConsentID}
RecordOnBlockchain(EnrollmentTX)
// Step 5: Return Transaction Receipt
Return EnrollmentTX
End Function
```

The above procedure registers the customer's metadata and uploads encrypted documents to IPFS. A unique transaction is recorded on the blockchain of the host shard, ensuring privacy, security, and traceability.

4) Cross-Shard Enrollment

After adding the customer's data to the host shard, the host shard sends the enrollment record to other relevant shards. Then, other shards verify the data and store a reference or replicate the record, and send an acknowledgment back to the host shard to confirm successful synchronization. Below shows the procedure of cross-shard enrollment.

Algorithm 3: Cross-shard enrollment
Function ShareToShard(CustID, Metadata, CID, ConsentID)

```
For Each ShardID in ShardNetwork:
  SendToShard(ShardID, CustID, Metadata, CID, ConsentID)
  Response = AwaitAcknowledgment(ShardID)
    If Response != "Success":
       LogError(ShardID)
End Function
```

Phase 4: Encryption phase

This phase describes the encryption and decryption processes used for securely storing customer data and ensuring its integrity within the e-KYC system. The encryption process secures data before uploading it to the blockchain. The encryption process uses public parameters (params), specific attributes from the customer data (*M*), the secret key of FI ($PrivKFI_{ID}$), and the delegate key ($PrivDKFI_{ID}$). The resulting metadata is encrypted, hashed, and uploaded to the blockchain as part of the transaction. SC4 performs encryption through the following steps:

Step 1: Compute a random value r used to generate a cryptographic commitment tied to metameta:.

$$r = H_1(\ PrivDKFI_{ID} \parallel M)$$

Step2: Compute a point on the elliptic curve using r and the generator point *P:*

$$R\ = r \cdot P$$

Step 3: Derive a shared secret using R and $PrivDKFI_{ID}$ to encrypt the metadata:

$$C_{FI_{ID}} = M \oplus H_1(R \parallel PrivDKFI_{ID})$$

Step 4: Compute a hash of $C_{FI_{ID}}$ and $meta$ for tamper-proofing.

$$h_{FI_{ID}} = H_1(C_{FI_{ID}} \parallel \mathrm{M})$$

Step 5: Derive signature components Sig_{FIID} to bind $PrivKFI_{ID}$ and r

$$\mathrm{Sig_{FIID}} = r - h_{FI_{ID}} \cdot PrivKFI_{ID}$$

Step 6: Combine all components into a hashed transaction $CustomerTX$ and upload it to Blockchain.

$$CustomerTX = H_1(NatID,\ Consent, CID,\ h_{FI_{ID}}\ ,\ Sig_{FIID})$$

Phase 5: Decryption Phase

In this phase, there are two major decryption parts: transaction decryption transaction and customer data decryption CT_M:

1) Transaction Decryption process Before decrypting M**,** the host FI runs the decryption algorithm as follows:

(1) Uses unique identifier such as $NatID$ to locate the corresponding $CustomerTX$ on the Blockchain. To extract the necessary field CID, it is done through:

$$CustomerTX = \{NatID, Consent,\ CID,\ h_{FI_{ID}}\ ,\ Sig_{FIID}\}$$

Then,

(2) Verifies the transaction Integrity by recomputing hash for validation.

$$h_{FI_{ID}}{}' = H_1(C_{FIID} \parallel M)$$

Check if

$$h_{FI_{ID}}{}' = h_{FI_{ID}}$$

If the hashes don't match, terminate the process as the metadata is invalid or has been tampered with.

2). Customer's Data Decryption

To decrypt *M*, the algorithm:

(1) Use the CID extracted from *CustomerTX* to locate and retrieve the encrypted data from the FI's IPFS node:

$$IPFS(CT_M) \rightarrow CID$$

(2) Decrypt the retrieved encrypted data using the symmetric key (SymKey):

$$M = \mathrm{DEC_{AES}}(SymKey, CT_M)$$

Phase 6: e-KYC process

In this phase, the system facilitates secure and privacy-preserving data sharing between FIs using ZKP and encryption. The process includes interactions between the source FI (FI_{source}) and the requesting FI (FI_{req}) to verify and share e-KYC data. The proposed system employs the ZK-Rollup technique for efficient proof aggregation and leverages sharded blockchain architecture for distributed proof verification. There are five steps in this phase.

Step 1: System Setup

Initialize cryptographic parameters, shard configurations, and inter-FI trust.

1.1). Trusted Setup for ZKP

To enable ZKP, zk-SNARKs is used for succinct proofs within a trusted or transparent setup. During this process, proving keys pk and verifying keys vk are generated as:

$$\{pk,\ vk\} \leftarrow Setup(\lambda)$$

pk and vk are then published for use by FIs and customers.

1.2). Shard Assignment

Users are divided into multiple shards, represented as:

$$S_1,\ S_2,\ \ldots\ ,S_n$$

Each shard is managed by a verifier node (VN_i), and each shard maintains its own local Merkle tree to store user proofs securely.

1.3). Inter-FI Key Exchange

To establish secure data sharing between financial institutions, FI_{source} and FI_{req} exchange their public keys:

$$PubK_{FI_{source}}, PubK_{FI_{req}}$$

1.4). Public Parameter**s**

The final step involves publishing public parameters that define the cryptographic and structural setup of the system. These parameters are represented as:

$$params\ = \{G,\ q,\ P,\ H,\ pk,\ vk,\ n\}$$

Step 2: User Authentication with source FI

This step verifies the customer with FI_{source} using ZKP and shard-based verification. The algorithm below describes the ZKP generation and verification process.

Algorithm 4: Proof Generation and Shard-Based Verification
Input: *CustID, PrivK$_C$, params*
Output: *Proof π_C*

Procedure GenerateProof(CustID, PrivK$_C$, params):
r ← random value from Z_c^*
*R ← r * P*
*c ← **H**(CustID || R || params)*
*s ← (r + c * PrivK$_C$) mod q*
π_C ← (CustID, R, s)
***Return** π_C*

//Procedure VerifyProof(π_C, PubK$_C$):
***Input:** π_C = (CustID, R, s), PubK$_C$*
***If** (s * P == R + c * PubK$_C$) then*
a. Add π_C to VN$_i$'s Merkle tree
b. Return "Valid Proof"
Else
c. Return "Invalid Proof"

//Procedure UpdateMerkleTree(π_C):
***Add** π_C to Merkle tree*
Compute new shard root: root$_i$ ← H(R$_1$ || R$_2$ || ... ||R$_m$)
***Return** root$_i$*

//ProcedureAggregateZK$_{rollup}$():
*Aggregate **all** shard proofs **into** ZK-Rollup proof π_{Rollup}*
*Submit (root$_{global}$, π_{Rollup}) **to** blockchain*

//Main Execution:
1. π_C ← GenerateProof(CustID, PrivK$_C$, params)
2. result ← VerifyProof(π_C, PubK$_C$)
*3. **If** result == "Valid Proof" then*
a. root$_i$ ← UpdateMerkleTree(π_C)
b. AggregateZK$_{rollup}$()
*4. **End***

Step 3: Customer Request for Data Sharing
The customer requests FI_{source} to share their data with FI_{req}.
1. Consent Submission (Customer): The customer creates and encrypts a consent message:
Consent = {ConsentID, CustID, FI_{source}, FI_{req}, Timestamp, CustSig}

$$Enc(Consent) = ENC_{RSA}(PubK_{FI_{source}}, Consent)$$

The encrypted consent is sent to FI_{req}.

2. FI_{req} Authentication: The customer authenticates with FI_{req} using π_C. FI_{req} verifies the proof and forwards Enc(Consent) to FI_{source}.

Step 4: FI_{source} Data Preparation
FI_{source} prepares and re-encrypts customer data for sharing. The procedure of this step is as follows:

1. Consent Validation: FI_{source} decrypts Enc(Consent) and validates the customer's signature:

$$Consent = DEC_{RSA}(PrivK_{FI_{source}}(Enc(Consent))$$

2. Data Retrieval: FI_{source} retrieves the encrypted customer data from IPFS:
{Enc(M), Enc(SymKey)} = Blockchain.query(CustID)

3. Data Re-encryption: FI_{source} re-encrypts the symmetric key (SymKey) for FI_{req}:
$$Enc(SymKey) = ENC_{PubK_{FI_{req}}}(SymKey)$$

4. Transformation Key Generation: A transformation key ($tk_{source \to req}$) is generated:

$$tk_{source \to req} = H_3(meta \,\|\, r \cdot PrivDK_{FI_{source}} \cdot PubK_{FI_{source}}) \oplus H_3(meta \,\|\, r \cdot PubK_{FI_{req}})$$

5. Transformed Ciphertext Creation: FI_{source} transforms the metadata and outputs:
$$CT'_{meta} = \{C_{FI_{req}}, meta, pid_{FI_{req}}, h_{FI_{source}}, Sig_{FI_{source}}\}$$

Step 5: Data Transfer to FI_{req}
FI_{source} securely transfers the encrypted data to FI_{req}. The procedure of this step is as follows:

1. Data Transmission: FI_{source} sends the following to FI_{req}:
- Transformed ciphertext (CT'_{meta}).
- Re-encrypted symmetric key (Enc(*SymKey*)).
- Encrypted PII file (ENC(*M*)).
2. Proof Submission: FI_{source} submits a ZKP ($\pi_{source \to req}$) to the blockchain to prove the transformation process.

Step 6: FI_{req} Verification and Decryption
FI_{req} verifies and decrypts the received data through the following procedure.
1. Symmetric Key Decryption: FI_{req} decrypts the symmetric key:
$$SymKey = DEC_{RSA}(PrivK_{FI_{req}}(Enc(SymKey))$$

2. PII Decryption: ***FI_{req}*** decrypts the PII file:
$$M = DEC_{AES}(SymKey, ENC(M))$$

3. Metadata Verification: FI_{req} verifies the integrity of the metadata using the transformation key ($tk_{source \to req}$).

4. Data Authenticity Check: FI_{req} checks the hash value and signature from FI_{source} to ensure the data's integrity and authenticity.

Phase 7: Optimized Redaction Phase
The redaction phase leverages off-chain processing for scalable

and efficient data modification and dynamic key rotation using threshold cryptography to enhance security. The following algorithm details the steps to implement the optimized redaction phase in which the updated Chameleon Hash is computed off-chain. in compliance with privacy regulations such as GDPR. This phase is executed by the Redaction smart contract. There are five steps as follows:

Step 1: Validate Redaction Request: The Redaction smart contract verifies whether the transaction exists in the blockchain. If it exists, the customer's consent and the customer's digital signature are verified through the following functions.

Exists (TX_{ID}) = True
If TX_{ID} in Blockchain
Else False
VerifySignature(Consent, CustSig, $Pubk_{Cust}$) = True

Step 2: Retrieve Chameleon Hash:
The smart contract retrieves the original transaction, including the data (M), randomization factor (r), and the Chameleon Hash (H_{CH}). In this step, r is decrypted off-chain using the trapdoor key shares.

Retrieve transaction components:
$T_x = \{M, r, H_{CH}(M, r)\}$

-Decrypt the randomization factor:
r = Decrypt($Tdk_1, Tdk_2, ..., Tdk_t$)

Step 3: Generate Controlled Collision:
A new randomization factor (r') is generated, and the updated Chameleon Hash is computed off-chain. The hash consistency is verified to ensure integrity.

Generate a new randomization factor:
r' = GenerateRandom()

Compute the updated Chameleon Hash:
$H_{CH}(M', r')$ = H(M') + f(r', $PubK$)
Ensure: $H_{CH}(M', r') = H_{CH}(M, r)$

Step 4: Key Rotation (Optional):
In this step, we introduce the Key rotation method using Threshold cryptography to ensure long-term security by periodically updating the trapdoor key (Tdk).

Threshold cryptography splits the current trapdoor key into shares ($Tdk_1, Tdk_2, ..., Tdk_n$) distributed among n participants. A threshold t of these shares is required to reconstruct the key for secure rotation. The new key is then generated, split into shares, and securely redistributed based on Shamir's Secret Sharing.

The below pseudocode presents the key rotation procedure:

Algorithm 5: Redaction

```
def rotate_trapdoor_key(current_shares, threshold):

Input:
-current_shares: List of trapdoor key shares {Tdk_1, Tdk_2, ..., Tdk_n}
-threshold: Minimum number of shares required to reconstruct T_d (denoted as t)
-n: Total number of participants

 Outputs:
- new_shares: New trapdoor key shares {Tdk_new,1, Tdk_new,2, ..., Tdk_new,n}
- Tdk_old: Archived old trapdoor key

Procedure rotate_trapdoor_key(current_shares, threshold):

//Step 1: Agreement Phase
agreeing_shares←collect_agreeing_shares(current_shares, threshold)
If size(agreeing_shares) < threshold Then
        Return "Threshold not met for key rotation"
End If

//Step 2: Reconstruct Current Key
T_d ← reconstruct_key(agreeing_shares)
T_d ← Combine({Tdk_1, Tdk_2, ..., Tdk_t})


//Step 3: Generate New Key
Tdk_new ← generate_random_key()

//Step 4: Create New Shares
1.new_shares ← Empty List
2.coefficients ← [Tdk_new] + [random_number(1, prime_field) for i in range(threshold - 1)]
  For i from 1 to n Do
      x ← i
     y ← Sum(coeff * (x ^ power) for power, coeff in coefficients) mod prime_field
      new_shares.append((x, y))

End For

//Step 5: Secure Share Distribution
    For each participant, share in (participants, new_shares) Do
       encrypted_share ← encrypt_with_public_key(participant.public_key, share)
        send_to_participant(participant, encrypted_share)
End For

//Step 6: Archive Old Key
    archive_key(T_d)
   return new_shares, Tdk_new
```

Step 5: Audit and Logging:
An off-chain audit log records the redaction event. A hash of the audit log is stored on-chain for integrity and transparent

V. Evaluation

A. Functionality Comparison

Table 1. Comparison of system function

Scheme	F1	F2	F3	F4
[2]	✗	✓	✗	✗
[6]	✗	✓	✗	✗
[11]	✓	✓	✓	✗
[13]	✓	✓	✓	✗
Ours	✓	✓	✓	✓

Note: F1= Anonymous Authentication, F2= Scalability Improvement F3= secure data portability, F4= Redaction

In this functionality comparison, we evaluate our proposed scheme against existing works based on four key functionalities: anonymous authentication, scalability improvement, secure data portability, and redaction. While some existing schemes incorporate a subset of these features, they exhibit limitations that prevent them from offering a fully secure, adaptable, and comprehensive solution. In [2] focuses on scalability improvement but lacks anonymous authentication, secure data portability, and redaction, making it unsuitable for secure identity verification. Without authentication, unauthorized access is a risk, and the absence of redaction and data portability limits its flexibility in handling and modifying sensitive data securely.In [6] provides scalability improvement but lacks secure data portability and redaction, restricting its usability in applications requiring strict privacy regulations and secure data management.In [11] and [13] offer authentication and scalability, ensuring some level of security and efficiency, but fail to include redaction, which is crucial for GDPR compliance and secure information management. Without redaction, these schemes cannot modify or erase stored data securely while maintaining blockchain integrity, limiting their adaptability for evolving data protection requirements.

In contrast, our proposed system fully integrates all four key functionalities, providing a scalable, secure, and privacy-preserving architecture. By combining redaction, authentication, and secure data portability, our solution ensures efficient identity management, compliance with data protection laws, and enhanced security, making it a robust choice for decentralized applications and secure digital ecosystems.

B. Communication Cost Analysis

Our proposed scheme, which integrates sharded blockchain architecture with ZK-Rollup, significantly optimizes communication costs compared to traditional blockchain-based e-KYC systems. In conventional blockchain architectures, each proof verification, metadata encryption, and transaction logging require separate on-chain storage and processing, leading to O(n)communication complexity in scenarios with multiple financial institutions (FIs) and high-volume transactions. Each FI must independently verify and store authentication proofs, metadata hashes, and encrypted customer data, which increases the amount of data exchanged and stored on-chain.In contrast, our scheme leverages sharded blockchain structure to distribute authentication requests across multiple shards, reducing congestion and localizing data processing, thereby improving scalability. The integration of ZK-Rollup further compresses multiple proofs into a single succinct proof before submission to the blockchain, reducing the size of transmitted proofs from $O(n)$ to $O(1)$. This means that instead of each authentication proof being verified separately on-chain, all proofs within a batch are aggregated and verified in constant time, minimizing communication overhead and reducing storage costs. Additionally, metadata encryption is performed using a lightweight commitment scheme, ensuring that only hashed identifiers and minimal metadata are stored on-chain, further reducing the size of blockchain transactions.

C. Computation Cost Analysis

(1) Blockchain Redaction Cost

In addition to the comparison of e-KYC related computation cost, we also provide the blockchain redaction computation cost analysis between our scheme and recent blockchain redaction approaches including scheme [14] scheme [15], and scheme [21]. Table 3 depicts the redaction cost comparison. We define the following notations for the analysis.

C_{CH}: Chameleon Hashing Cost
C_{CH_Col}: Collision Generation Cost
C_{ECC_Dec}: Cost of ECC decryption
t: threshold (out of n participants)
C_{ModExp}: Cost of modular exponentiations
|A|: No of Attributes in the Access Policy
$C_{CP\text{-}ABE\_Dec}$: Cost of Attribute-based Decryption
C_{PE_Update}: Cost of ciphertext update based on puncturable encryption

Table 2: Redaction Cost Comparison

Operation	Scheme [13]	Scheme [14]	Scheme [21]	Ours
Chameleon Hash Computation	C_{CH}	C_{CH}	C_{CH}	C_{CH} *(Off-Chain)*
Collision Generation for Redaction	C_{CH_Col}	C_{CH_Col}	C_{CH_Col}	C_{CH_Col} *(Off-chain)*
Key Usage for Redaction	$C_{ECC\_Dec} + C_{Hash}$	$O(n)*(C_{ECC\_Dec} + C_{Hash})$	$O(\|A\|)+C_{CP\text{-}ABE\_Dec}+C_{CH}+C_{PE\_Update})$	$O(t)*(C_{ECC\_Dec}+C_{ModExp})$

As shown in Table 2, our scheme significantly optimizes blockchain redaction costs compared to previous works by offloading Chameleon Hash computation and collision generation to off-chain processing, thereby reducing the on-chain storage and computation burden. Unlike Scheme [13], [14], and [21] which perform on-chain Chameleon Hash and collision generation, our scheme delegates these computations off-chain, thereby minimizing blockchain transaction overhead and improving system scalability. A critical cost factor is the key usage for redaction, where different schemes employ varying cryptographic mechanisms. Scheme [13] follows a single-entity redaction model, requiring only ECC decryption (C_{ECC_Dec}) and hashing (C_{Hash}) for redaction key retrieval, making it relatively lightweight but lacking multi-party security guarantees. Scheme [14] incorporates multi-party redaction, leading to an increased computational cost of $O(n)*(C_{ECC\_Dec} + C_{Hash})$, where n is the number of participating entities involved in the redaction process.

In contrast, Scheme [21] leverages CP-ABE decryption ($C_{CP\text{-}ABE\_Dec}$) and puncturable encryption (C_{PE_Update}) for fine-grained access control, allowing for dynamic key revocation. However, this introduces higher computational overhead, as each redaction request requires $O(|A|) + C_{CP\text{-}ABE\_Dec} + C_{PE\_Update}$, where |A| represents the number of attributes in the access policy. The necessity to re-encrypt the trapdoor key (Tdk) and update previous ciphertexts results in substantial computational costs, particularly in scenarios with frequent redaction requests. Our scheme adopts a threshold cryptography-based redaction approach, which distributes key shares among participants, ensuring that redaction is securely authorized without requiring all participants to be online. By leveraging Shamir's Secret Sharing, our scheme reduces the redaction key retrieval cost to $O(t)*(C_{ECC\_Dec} + C_{ModExp})$, where t is the threshold number of required participants. Compared to Scheme [21], our approach eliminates the need for complex ciphertext updates CP-ABE decryption, making it significantly more efficient for large-scale GDPR-compliant redaction requests.

D. Computation Cost Analysis

(2) e-KYC Authentication, Data Encryption/Decryption, and Integrity verification Cost

To demonstrate the efficiency of the core e-KYC processes, we compare the computation cost related to the e-KYC user and credential authentication , data encryption/decryption cost , and integrity verification through Merkle Tree method of our scheme and three e-KYC solutions including scheme [2], [6], and [13]. Table 3 presents the comparison of e-KYC computation cost comparison.

Table 3. Comparison of the computation cost

The following notations are used to analyze the e-KYC- related computation cost.

$C_{RSASign}$: RSA signing/verifiction cost
$C_{ECCSign}$: ECDA signing/verification Cost
C_{sym}: 256-bit AES Encryption/Decryption cost
C_{pub}: Public key encryption cost based on 1024-bit RSA
|N|: No. of Nodes in the CP-ABE based Access Polity
C_{Exp}: Exponentiation and XOR operation cos
C_m : Multiplication operation cost
C_{PQC}: Post Quantum Computing Encryption/Decryption Cost $O(n^2)$ for lattice-based encryption
C_{QKD} : Quantum Key Distribution Setup having the complexity $O(n)$ for n network nodes.
C_{QKA}: Quantum Key Agreement Encoding having the complexity $O(\log n)$
$C_{QnetGen}$: Quantum Network Access Control setup cost having the complexity $O(\log n)$
C_{Qver} : Quantum Key Agreement Verification Cost having the complexity $O(\log n)$
$C_{QnetVer}$: Quantum Network Access Control Verification cost having the complexity $O(\log n)$
$C_{QHashVer}$: Quantum hashing verification cost having the complexity $O(\log n)$
C_{ECC} : ECC Encryption/Decryption cost
$C_{ECCScaMul}$: the cost of an **elliptic curve scalar multiplication.**
$C_{ECCPointAdd}$: the cost of an **elliptic curve point addition.**
C_{Hash} : the cost of a **cryptographic hash function.**
C_{ModAdd} : the cost of a **modular addition.**

Table 3: Comparison of Computation Cost

Scheme	Authen Gen/ Proof Gen	AuthenVer/ Proof Ver	Data Enc	Data Dec	Re-Encryption(data transfer across FIs)	Merkle Tree Proof Checking
[2]	$C_{RSASign}$	$C_{RSASign}$	$C_{sym}+C_{Pub}+(2+2\|T\|)C_{exp}+C_m$	$C_{sym}+\|N\|C_{exp}+(1+2\|S\|)C_P+\|N\|C_P$	N/A	O(n)
[6]	$C_{QKD}+C_{QKA}+C_{QNetGen}$	$C_{Qver}+C_{QnetVer}+C_{QHashVer}$	C_{PQC}	C_{PQC}	N/A	O(log n)
[13]	$C_{ECCSign}+(2*C_{ECCScaMul})+C_{Hash}+C_{ModAdd}$	$C_{ECCSign}+(2*C_{ECCScaMul})+C_{Hash}+C_{ModAdd}$	$C_{ECCScaMul}$	$C_{ECCScaMul}$	N/A	O(n)
Ours	$C_{ECCScaMu}+C_{Hash}+C_{ModAdd}$	$C_{ECCScaMu}+C_{Hash}+C_{ModAdd}$	C_{sym}	C_{sym}	C_{ECC}	O(log n)

Based on the cost details presented in Table 3, our comparison highlights key trade-offs between security, efficiency, and scalability across authentication and encryption methods. Traditional **RSA-based authentication** [2] incurs high computational costs due to **public key encryption and** RSA signing cost, making it inefficient for large-scale e-KYC applications. Quantum-based authentication [6], leveraging Quantum Key Distribution and Quantum Key Agreement, ensures post-quantum security but suffers from $O(n^2)$ encryption and $O(n)$ setup complexity, making it impractical for widespread adoption. In contrast, ECC-based authentication [13] balances security and efficiency by using ECC scalar multiplication, hashing, and modular addition, reducing overhead while maintaining strong security. However, ECC-based encryption and decryption remain more expensive than symmetric key approaches. Our proposed scheme optimizes authentication and encryption with lightweight ECC-based signing, AES encryption, and efficient ECC re-encryption for secure data transfer. Authentication verification leverages $O(\log n)$ Merkle tree proof checking, integrating sharded blockchain and ZK-Rollup for scalable proof aggregation. Moreover, only our scheme has ability to securely transfer encrypted customer data between FIs through ECC-based re-encryption further ensures efficiency in inter-bank transactions while maintaining data privacy and security. Accordingly, compared to current e-KYC approaches, our approach ensures efficient authentication, low-cost encryption, and scalable inter-FI data sharing, making it well-suited for real-world e-KYC applications.

REFERENCES

[1] General Data Protection Regulation (GDPR), https://gdpr-info.eu, Accessed 8 Jan 2025

[2] S. Fugkeaw, "Enabling Trust and Privacy-Preserving e-KYC System Using Blockchain," in *IEEE Access*, vol. 10, pp. 49028-49039, 2022, doi: 10.1109/ACCESS.2022.3172973.

[3] Y. Suga, "ICH 3-party model for claims distribution on the blockchain and their application to the e-KYC-e model," 2022 IEEE
4th Global Conference on Life Sciences and Technologies (LifeTech), Osaka, Japan, 2022, pp. 512-513, doi: 10.1109/LifeTech53646.2022.9754919.

[4] B. Jaber, D. Kriwiesh and M. A. AlRagheb, "A Blockchain Framework in the Banking Sector Based in e-KYC System Conceptual Framework," 2024 2nd International Conference on Cyber Resilience (ICCR), Dubai, United Arab Emirates, 2024, pp. 1-4, doi: 10.1109/ICCR61006.2024.10532852.

[5] Feulner, S., Schlatt, V., Guggenberger, T., Völter, F., Stoetzer, JC. (2024). Self-Sovereign Identity for Digital KYC. In: Fridgen, G., Guggenberger, T., Sedlmeir, J., Urbach, N. (eds) Decentralization Technologies. Financial Innovation and Technology. Springer, Cham. https://doi.org/10.1007/978-3-031-66047-4_7

[6] G. M. Faruk Ahmed, Imran Hasan, Md Soumike Hassan, Rahul Khan, Abu Jor Al Gefari, Zain Buksh, Joseph Liu, A B M Shawkat Ali, "Enhancing e-KYC Security and Privacy: Harnessing Quantum Computing and Blockchain in Web 3.0", Distributed Ledger Technologies: Research and Practice, 2024, doi/10.1145/3686166.

[7] Pradnya Patil, M. Sangeetha, Blockchain-based Decentralized KYC Verification Framework for Banks, Procedia Computer Science,Volume 215, 2022, Pages 529-536.

[8] M. Kumar, P. A. Nikhil, and P. Anand, “A blockchain based approach for an efficient secure kyc process with data sovereignty,” Int J Sci Technol Res, vol. 9, pp. 3403–3407, 2020

[9] A. A. Mamun, S. R. Hasan, M. S. Bhuiyan, M. S. Kaiser and M. A. Yousuf, "Secure and Transparent KYC for Banking System Using IPFS and Blockchain Technology," *2020 IEEE Region 10 Symposium (TENSYMP)*, Dhaka, Bangladesh, 2020, pp. 348-351, doi: 10.1109/TENSYMP50017.2020.9230987.

[10] S. Bhatia, L. Vishwakarma and D. Das, "Cost-efficient Blockchain-based e-KYC Platform using Biometric verification," *2024 International Conference on Information Networking (ICOIN)*, Ho Chi Minh City, Vietnam, 2024, pp. 403-408, doi: 10.1109/ICOIN59985.2024.10572111.

[11] Schlatt, V., Sedlmeir, J., Feulner, S., & Urbach, N. (2022). Designing a framework for digital KYC processes built on blockchain-based self-sovereign identity. Information & Management, 59. https://doi.org/10.1016/j.im.2021.103553

[12] Patkar, M., Giri, A., Gite, P., Shinde, A., Shetye, K. (2024). Privacy Preserving and Trustworthy E-KYC System Using Blockchain. In: Mishra, D., Yang, X.S., Unal, A., Jat, D.S. (eds) Data Science and Big Data Analytics. IDBA 2023. Data-Intensive Research. Springer, Singapore. https://doi.org/10.1007/978-981-99-9179-2_49

[13] K. Takaragi, T. Kubota, S. Wohlgemuth, K. Umezawa, and H. Koyanagi. 2023. Secure Revocation Features in eKYC-Privacy Protection in Central Bank Digital Currency. In *IEICE Transactions on Fundamentals of Electronics, Communications, and Computer Sciences*, 106, 3 (2023), 325–332.

[14] L. -Y. Yeh, W. -H. Hsu and C. -Y. Shen, "GDPR-Compliant Personal Health Record Sharing Mechanism With Redactable Blockchain and Revocable IPFS," in *IEEE Transactions on Dependable and Secure Computing*, vol. 21, no. 4, pp. 3342-3356, July-Aug. 2024, doi: 10.1109/TDSC.2023.3325907.

[15] G. Zhou, X. Ding, H. Han and A. Zhu, "Fine-Grained Redactable Blockchain Using Trapdoor-Hash," in *IEEE Internet of Things Journal*, vol. 10, no. 24, pp. 21203-21216, 15 Dec.15, 2023, doi: 10.1109/JIOT.2023.3279434.

[16] J. W. Heo, G. Ramachandran and R. Jurdak, "Decentralised Redactable Blockchain: A Privacy-Preserving Approach to Addressing Identity Tracing Challenges," *2024 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, Dublin, Ireland, 2024, pp. 215-219, doi: 10.1109/ICBC59979.2024.10634438.

[17] H. Guo, W. Li, C. Meese and M. Nejad, "Decentralized Electronic Health Records Management via Redactable Blockchain and Revocable IPFS," *2024 IEEE/ACM Conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE)*, Wilmington, DE, USA, 2024, pp. 167-171, doi: 10.1109/CHASE60773.2024.00028.

[18] Yueyan Dong, Yifang Li, Ye Cheng, Dongxiao Yu,
Redactable consortium blockchain with access control: Leveraging chameleon hash and multi-authority attribute-based encryption, High-Confidence Computing,Volume 4, Issue 1,
2024,100168

[19] Y. Zhang, Z. Ma, S. Luo and P. Duan, "Dynamic Trust-Based Redactable Blockchain Supporting Update and Traceability," in *IEEE Transactions on Information Forensics and Security*, vol. 19, pp. 821-834, 2024, doi: 10.1109/TIFS.2023.3326379.

[20] L. -E. Wang, Y. Wei, P. Liu and X. Li, "Secure and Trusted Copyright Protection for Educational Data on Redactable Blockchains," *2023 IEEE 29th International Conference on Parallel and Distributed Systems (ICPADS)*, Ocean Flower Island, China, 2023, pp. 617-622, doi: 10.1109/ICPADS60453.2023.00096.

[21] S. Xu, J. Ning, X. Li, J. Yuan, X. Huang and R. H. Deng, "A Privacy-Preserving and Redactable Healthcare Blockchain System," in *IEEE Transactions on Services Computing*, vol. 17, no. 2, pp. 364-377, March-April 2024, doi: 10.1109/TSC.2024.3356595