Open access • Journal Article • DOI:10.1002/CPE.5520

Towards a secure incremental proxy re-encryption for e-healthcare data sharing in mobile cloud computing — Source link

Tarunpreet Bhatia, Anil Kumar Verma, Gaurav Sharma

Institutions: Thapar University, Université libre de Bruxelles

Published on: 10 Mar 2020 - Concurrency and Computation: Practice and Experience (John Wiley & Sons, Ltd)

Topics: Mobile cloud computing, Cloud computing, Proxy re-encryption and Data sharing

Received: 16 June 2018 | Revised: 25 March 2019 | Accepted: 30 May 2019
DOI: 10.1002/cpe.5520

WILEY

RESEARCH ARTICLE

Towards a secure incremental proxy re-encryption for e-healthcare data sharing in mobile cloud computing

Tarunpreet Bhatia[1] | A.K. Verma[1] | Gaurav Sharma[2]

[1]Department of Computer Science and Engineering, Thapar Institute of Engineering and Technology, Patiala, India
[2]Département d'Informatique, Université libre de Bruxelles, Bruxelles, Belgium

Correspondence
Tarunpreet Bhatia, Department of Computer Science and Engineering, Thapar Institute of Engineering and Technology, Patiala-147 004, India.
Email: tarunpreetbhatia@gmail.com; tarunpreet@thapar.edu

Summary

Cloud computing provides universal access to a pool of shared resources to numerous stakeholders/shareholders of the e-healthcare industry. The speedy adoption of cloud computing has inevitably raised security concerns for the outsourced data. Since mobile devices are resource constrained, the security solutions must discharge the computing comprehensive operations on the cloud for implementation. Conventionally, any modification to uploaded record would compel the mobile client to encrypt and compute the hash value from scratch. Through this paper, we intend to propose a pairing-free incremental proxy re-encryption scheme, without certificates, which would run proportionate to the number of modifications in time, instead of the document length for improvement in the file modification tasks. The proposed scheme shows a significant improvement in the file modification system regarding the energy consumption and the turnaround timer taken. The proposed scheme has been verified through a formal method using Z_3 solver.

KEYWORDS

cloud computing, e-healthcare, incremental cryptography, mobile Cloud, proxy re-encryption, security

1 | INTRODUCTION

The life-saving efforts in the eleventh hour have often fallen hard back even with advancements in the health care industry and medical services. One of the reasons for such situation is the unavailability of the patient's records that are generally stacked in the piles of paperwork. Firstly, these records are highly ambiguous with multiple gaps in information. Secondly, these records are generally scattered at several heath care centers through the region with different physicians, pharmacies, and hospitals. A number of information systems that contribute to the development of a more complete electronic health record (EHR) are being implemented by the hospitals nowadays to address these problems. Information regarding the patient's health care can be accessed instantly with the help of the electronic systems. An EHR is a centralized source storing medical and treatment history of patients' health information. These valuable patient-centered records can be accessed easily by all the approved healthcare providers (laboratories, specialists, pharmacists, etc).

E-health applications generate a lot of data that become difficult for the health care units to store on a single computer because of catastrophic failures. Cloud services thus provide techniques for outsourcing the restrictive expensive computations and data in a secure manner, and this is prevalent in the research community.[1-6] Muhameed et al[7] proposed a ubiquitous health care framework (UbeHealth) encompassing edge computing, Internet of things (IoT), high-performance computing to address the challenges including network latency, deep learning, reliability, big data, bandwidth requirement, rising costs, etc, for satisfying next-generation health care needs.

The clients owning resource-constraint mobiles can outsource the heavy computations and massive data onto the suspicious cloud servers and benefit from the unlimited computing and storage services in a pay-per-use manner. Despite the inherent advantages of cloud computing in e-healthcare industry, the herculean task of migrating e-health data on the public cloud impedes its widespread adoption. A crucial aspect to be addressed concerning the processing and storage of data on cloud is to safeguard the confidentiality and integrity of the patient's medical information.[8] The concerns regarding the confidentiality and integrity of patients' data is of utmost priority as any mischievous modification on EHRs could result in erroneous health decisions. Although the role played by cloud computing is intermediary, the health care institutes

employing such services are vulnerable as they lose the ownership, control, or management of the EHR data, which are indirectly processed and stored on suspicious servers. Any malicious intentions on part of the cloud service providers (CSPs) would irrevocably endanger the reliability of the patients' medical data records.

EHRs are distinguished from generic cloud data via their characteristic set of operational constraints, security, and regulatory.

- EHRs must adhere to the Health Insurance Portability and Accountability Act (HIPAA), General Data Protection Regulation, and other security guidelines following its strict regulatory policies (administrative, technicalm and physical) for it to be law compliant. HIPAA mandates the protection and confidential handling of medical documents and personal health information.
- Individual EHRs contain massive data in the form of X-rays, MRIs, CT scans, pathology lab test reports, medical transcripts, and physician's diagnosis, etc, which keep on increasing over the life period of the respective patient. The security solution for e-healthcare data storage in cloud should work proficiently on large data sets.
- Individual EHRs needs to be modified often as per the medical history of the respective patients. The security solution for e-healthcare data storage in cloud should be capable of securely supporting the dynamic insert, delete, and update operations on cloud data.
- Presently, a major part of EHRs is examined and updated using low battery-powered mobile devices. It is imperative to design energy-efficient security solution for e-healthcare data storage in cloud for a judicious use of the battery resources of the energy-starved mobile devices operated by physicians, medical personnel, and the patients.
- EHRs should be accessible for longer periods, spanning for about an entire lifetime of the patient all the while maintaining its confidentiality and integrity of ever-increasing EHRs during entire detainment period.

The above-mentioned constraints render the design of security mechanism for EHRs quite intricate and challenging. Due to the sensitive nature of EHRs constituents, a mobile client should encrypt the e-healthcare data before outsourcing to cloud to prevent its exposure to intruders and CSPs. Any health care confidentiality and integrity protocol should target the regulatory and operational requirements and the EHR security to specify the migration of medical records to the cloud.

The digital universe is expanding just like the physical universe and may embrace nearly as many digital bits as there are stars in the universe by 2020 and will reach 44 zettabytes.[9] IoT Tsunami has created a mark for itself in the digital universe. The digital universe is purported to grow from a mere 3.8% in 2016 to 10% by 2020 with the inception of IoT and related technologies. The engulfing waves of IoT comprise tens of billions of sensors, billions of intelligent systems, and millions of applications flooding our memory system due to round-the-clock digital data acquisition and generation. According to Gartner, Inc, there will be nearly 26 billion IoT devices by 2020, which will demand smart industry solutions generating revenue in excess of $300 billion to IT vendors and service providers. According to Cisco recent forecast,[10,11] ‘‘annual global IP traffic passed the zettabyte threshold by the end of 2016 and will reach 2.3 ZB per year by 2020. Global mobile data traffic will increase sevenfold between 2016 and 2021. Mobile data traffic will grow at a compound annual growth rate (CAGR) of 47% from 2016 to 2021, reaching 49.0 exabytes per month by 2021.’’ Thus, the scale of data being processed every day by various cloud services, sensor networks, distributed storage systems, and digital media service demands for novel secure and efficient solutions based on the incremental cryptography paradigm. The advent of IoT has given place for better monitoring and surveillance of patients' health via the acquisition of their body parameters through small wearable and implantable sensors. The data collected are communicated over short-range wireless communication devices. The data captured by the sensor nodes need to be stored for further analysis. With the wider adoption of wireless body area networks (WBANs) and reduction in the cost of the sensors due to improvement in sensor technology, the data being generated and henceforth stored are expected to increase exponentially. Thus, WBAN finds itself marred in the security and privacy concerns regarding the storage of EHRs either within the domain of WBAN or outsourcing to third-party CSPs.

1.1 | Related work

The traditional cryptography approach (symmetric or asymmetric encryption) hinders flexible sharing of data with multiple users as it requires the data owner to share the keys for decryption. Proxy re-encryption (PRE) is a basic cryptographic primitive, which enables the re-encryption of ciphertext from one secret key to other without knowing the actual secret keys. Blaze et al[12] proposed an atomic re-encryption scheme based on El-Gamal cryptosystem.[13] Under re-encryption, the ciphertext encrypted with user $A's$ public key can be altered into ciphertext that is encrypted by user B's public key. The proxy re-encryption primitive offloads the re-encryption and data access–related tasks on to a trusted third-party proxy server, which remains oblivious of the content to be re-encrypted. In the evolving cloud computing infrastructure, relying on traditional integrity mechanisms such as message authentication codes (MACs) and digital signatures in their classical forms would not suit the constraints of health care services. The existing security solutions for cloud environment[14-16] neglected the resource-constrained nature of the mobile devices.

Jia et al[17] and Zhou et al[18] suggested data security schemes based on the notion of proxy re-encryption for mobile cloud environment by offloading the data-intensive operations from a mobile device to the cloud without exposing the security keys and data content to anyone. Yang et al[19] proposed modified public-verifiable data possession scheme for secure storing of data for resource-constrained mobile devices in cloud computing. It comprises of a trustworthy third-party auditor to perform security-related tasks such as encryption, decryption, signature generation, and verification. However, it relies on expensive bilinear mapping and Merkle hash trees for ensuring integrity.

Tysowski and Hasan[20] proposed a manager-based re-encryption (MReS) and cloud-based re-encryption (CReS) model to ensure data security in cloud without incurring much handling overhead on mobile devices. The scalability of MReS is restricted by the servicing manager, which becomes a bottleneck with the rush of read requests initiated by the mobile clients. In CReS, CSPs perform computation intensive data re-encryption tasks, while the mobile client is responsible for the overall key management-related tasks. An enhancement over MReS and CReS, cloud-manager–based re-encryption scheme (CMReS) was recommended by Khan et al.[21] Unlike the manager in MReS, the role of manager in CMReS is completely surpassed allowing the data owner to autonomously download and decrypt the data. The onus for generation of shared key pair lies with the mobile client while re-encryption related tasks are offloaded to cloud without revealing secret keys. The major shortcoming of CMReS is an expensive update operation for large data sets wherein mobile devices need to recompute and update the ciphertext from scratch without specifically focusing on the altered portion(s) of the file.

Mollah et al[22] proposed a protected data sharing and searching scheme by authorized users only which offloaded security operations to edge servers. The authors encrypted the data using secret key encryption (Advanced Encryption Standard), public key encryption (Rivest-Shamir-Adleman (RSA)), and hash functions (Security Hash Algorithm 256 (SHA-256)). Khan et al[23] proposed a workload distribution model for offering resource-intensive, re-encryption, and decryption tasks on trusted entity. It has been assumed that the communication channel between mobile devices, encryption service providers, and decryption service providers are secure. The workload is distributed between a mobile device and a trusted body as 10% minimum and 90% maximum. The message to be encrypted is split into two parts: one part for the mobile device and the other for the trusted entity.

Zhang et al[24] proposed privacy-aware secure health (PASH) attribute-based access-control system supporting large universe with hidden attribute values. PASH was proved to be secure in the standard model addressing both data security and user privacy. PASH provides attribute privacy and efficient decryption test in terms of bilinear pairing operations in comparison to existing schemes.[25,26] Ali et al[27] proposed SeSPHR for sharing of personal health records (PHRs) securely by utilizing semitrusted proxy server for generating public, private and re-encryption keys and ensuring access control. It relies on El-Gamal encryption and proxy re-encryption for providing data security and access control lists (ACLs) for different sets of users. However, they assumed communication between client and proxy server secure by the use of IP security or Secure Sockets Layer (SSL).

Bhatia et al[28] explored various proxy re-encryption schemes and proposed an efficient proxy re-encryption scheme, without certificates, as compared to existing schemes and proved its security in the random oracle model. Modi et al[29] proposed a hybrid approach using linear network coding for providing reliability, fault tolerance, and re-encryption based on ElGamal cryptography for securing health care data over the cloud.

Bellare et al[30,31] designed cryptographic algorithms founded on the notion of incremental cryptography that updates output efficiently as per the underlying input modifications. A file visualized as a sequence of blocks would undergo re-computations for generating the ciphertext, only for those blocks that underwent any alterations excluding the ones that remain unaltered.[32,33] If we apply the incremental cryptographic algorithm on a medical record and afterwards this record gets modified, it is feasible to update the result of the algorithm by only using the modified record blocks rather than recomputing the algorithm on the entire record with incremental cryptography. This contributes to major performance and energy effectiveness in e-healthcare setting because of the large size of medical documents and the frequent update mechanisms that are performed on them. Alongside the confidentiality aspect of EHRs, it is imperative to ensure their integrity, which would prevent their accidental or intentional modification.

Khan et al[34] suggested an incremental version of security schemes on resource-constrained mobile devices such as EnS, CoS, and ShS[14] to improve the block modification tasks. The mobile user provides a password that transforms into encryption and integrity keys to ensure data confidentiality. The progressive version consumes more resources of mobile devices during the early encryption and uploading phase. Itani et al[35] proposed integrity-enforcement protocol that leverages incremental cryptographic primitive and trusted computing and provides integrity of mobile users' EHRs stored on the cloud in mobile cloud computing (MCC) environment. However, confidentiality of the data is overlooked in this scheme. Khan et al[36] suggested first incremental proxy re-encryption scheme (I-PReS) as an extension of block-based sharing[16] that utilizes progressive cryptography for improving the file modification operations providing both integrity services and confidentiality.

The incremental hash functions have failed to make a mark in the industry due to following loopholes. Firstly, the known incremental hash functions[30,31] need to perform expensive modular operations over large prime integers to attain a certain level of security (for example, 2^{128} or 2^{256}), which makes their performance lag behind those of the simple cryptographic hash functions. Secondly, the level of security of the incremental hash functions is disproportional to the size of the calculated hash value by several thousand bits unlike ordinary cryptographic hash functions such as SHA-1,[37] SHA-2,[38] and SHA-3,[39] where claimed bit-security level of the hash function corresponds to the size of the calculated hash value. Incremental hashing can be achieved using Merkle trees,[40,41] which necessitate the storage of all the intermediate nodes' hash values.

In 2005, the attacks on SHA-1 performed by cryptanalysts proved the inappropriateness of the algorithm for future use,[42] and since 2010, many organizations have replaced it by SHA-2 or SHA-3. This was followed by subsequent announcements by Apple, Microsoft, Google, and Mozilla declaring SHA-1 SSL certificates unacceptable from 2017 onwards. CWI Amsterdam and Google successfully performed a collision attack against SHA-1 on February 23, 2017,[43] publishing two unrelated PDF files that gives the same SHA-1 hash value as an output. Mihajloska et al[44] proposed two incremental hash functions iSHAKE128 and iSHAKE256 for zettabyte era based on extendable-output functions (XOFs). iSHAKE128 and iSHAKE256 have the flexibility of generating an output of any desired length. Technically, XOF can be employed as a hash function by choosing

fixed output length, but the two outputs of a common message are closely related for two different outputs lengths, ie, hash output of longer length is an extension of shorter length, which is an undesirable property of hash functions.

1.2 | Motivation

A traditional way for including data confidentiality contains a significant cost overhead to the data owner in encryption of the data with public or shared symmetric key before outsourcing to the cloud for various data recipients.[28] PRE is a cryptographic scheme that allows Bob to delegate his decryption rights to another user Alice. In a PRE scheme,[12] a semihonest proxy agent of Bob re-encrypts the ciphertext encrypted under Bob's public key on behalf of Bob into ciphertext encrypted under Alice's public key without learning anything about the message and private key of Bob. The PRE schemes are useful in the scenarios where data are to be shared with the multiple authorized users over the cloud. Nevertheless, the integrity of EHRs should also be preserved, and usually, such EHRs should be signed by a trusted party. In any case, data hashing is inevitable, and as EHR is being updated, so the hash value needs to be recomputed to reflect the corresponding updations. The very context of medical data confines the modifications of EHRs to appendments of new data from sensors and minimal updates. An entire recalculation of the hash value on the updated EHR's sounds practically irrational if the size of the EHRs' is taken into perspective. Such modifications provide for an appropriate implementation of the incremental approach of data hashing. Incremental cryptography can minimize the overhead associated with recalculation of the hash value from the scratch by considering only modifications (insertion, update, and deletion) to a data set.

1.3 | Our contribution

One of the crucial factors for enabling fast and secure computations in the Zettabyte era is the use of incremental cryptographic primitives. For files ranging from several megabytes up to hundreds of gigabytes, incremental cryptographic primitives offer speedup factors measured in multiple orders of magnitude. There are only three security schemes[34-36] as per our knowledge, which applies the concept of incremental cryptography for MCC paradigm and only[36] includes proxy re-encryption with incremental cryptography. The scheme by Itani et al[35] enforces integrity only, and confidentiality of data is overlooked. The scheme[36] involves expensive bilinear pairing operations and downloading of intermediate MAC values of all blocks from cloud by a mobile user while conducting block insertion, deletion, or update operation, thereby increasing communication and storage overhead. It also suffers from a key escrow problem as proxy knows secret keys of all identities. The pairing-free incremental proxy re-encryption is the novelty of our manuscript. We intend to propose an incremental version of proxy re-encryption scheme for improving the file modification operations based on elliptic curves without certificates, in this paper. We employed recently proposed SHA-3 hash functions for incremental cryptography. We verified the proposed scheme using high-level Petri nets (HLPNs) and Z_3 solver.

1.4 | Road map

The rest of the paper is prepared as follows. Section 2 provides formal security model. The proposed CL-iPRE scheme is discussed in Section 3 followed by block modification operations in Section 4. Section 5 provides formal security analysis and verifies the correctness using Z_3 solver. Section 6 deals with simulation results. Finally, conclusions are formulated in Section 7.

2 | SECURITY MODEL

2.1 | Indistinguishability of incremental encryption

The secrecy requirements for incremental cryptography are as follows: Firstly, the basic elliptic-curve cryptography (ECC) encryption algorithm should be semantically secure. Secondly, the incremental modification step should not violate the semantic security of encryption algorithm and should not leak information about underlying modifications. In contrast to Bellare et al,[30] adversaries are allowed to know the location of modification but not the content that is modified.

2.1.1 | Security against type I adversary in game-I

In this section, a game is played between Type-I adversary Adv_1 and challenger. Type-I adversary is allowed to replace public keys of any entity but not access master secret key. The challenger maintains a list of public keys of identities and corrupted identities (whose secret key has been mined or public key has been replaced and partial key is extracted by adversary).

- Initialization and setup phase: System setup is run by the challenger with security parameter k and computes master secret key x and a list of public parameters δ. It gives δ to Adv_1 and keeps x secret.
- Phase-I: The challenger $\mathcal{A}dv_1$ can query partial key extract, public key extract, private key extract, public key replace, re-encryption key generate, and re-encryption oracles as in the work of Bhatia et al.[28] The challenger responds to all the oracles as mentioned by Bhatia et al.[28] In addition to these oracles, $\mathcal{A}dv_1$ can call incremental modification oracle with a location index and modifications as inputs. The challenger returns updated ciphertext and modified signature to the adversary.

- Challenge phase: Once $\mathcal{A}dv_1$ decides Phase-I of $Game_1$ is over, it yields target identity ID' and two messages EHR_0 and EHR_1 having equal number of blocks n or a ciphertext (C_A, C) with two modifications M_0 and M_1 of similar type and modifying same location (which adversary may know beforehand) on which it wishes to be challenged. A random bit $\beta \in \{0,1\}$ is chosen by challenger and kept secret from adversary. In the first case, EHR_β is encrypted. In the second case, incremental modification algorithm is applied to M_β and (C_A, C) and the result is given to the adversary $\mathcal{A}dv_1$.
- Phase-II: $\mathcal{A}dv_1$ is free to further call aforementioned oracles but may not submit the challenge ciphertext C.
- Guess phase: The challenger has to guess β to win the game trivially.

Definition 1. An incremental certificateless proxy re-encryption scheme $\left(t, q_{par}, q_{pri}, q_{pub}, q_{pub\_rep}\ q_{enc}, q_{re}, q_{re-enc}, \mu, q_{inc}, n; v\right)$ is chosen plaintext attack (CPA)–secure against indistinguishability if, for any time t, the adversary who makes at most q_{par} queries to partial key extract oracle, q_{pub} queries to public key extract, q_{pri} queries to private key extract, q_{pub_rep} queries to public key replace, q_{enc} queries to encryption oracles q_{re} queries to re-encryption key generate, q_{re-enc} queries to re-encryption oracles, and q_{inc} queries to incremental modification oracle (with μ as total blocks on which encryption and incremental modification oracles are called), we have advantage of adversary $\mathcal{A}dv_1$ less than v.

2.1.2 | Security against type II adversary in game-II

The master secret key can be accessed by Type II adversary but it is not allowed to replace public key. It can easily figure partial key of any entity so access to private key extract oracle is not required. The rest of the game and access to oracles is similar to Game-I.

2.2 | Unforgeability of ciphertext integrity of incremental encryption

In case of health care data sharing, incremental encryption is not sufficient. There is a need to ensure data integrity. An adversary should not be allowed to modify or forge ciphertext stored over an insecure medium so that it turned out to be a valid plaintext when decryption oracle is used.

2.2.1 | Security against type I adversary in game-I

A challenger and an adversary $\mathcal{A}dv_1$ participate in this game-I as follows:

- Initialization: System setup is run by the challenger with security parameter k and computes master secret key x and a list of public parameters δ. It gives δ to Adv_1 and keeps x secret.
- Queries: The adversary $\mathcal{A}dv_1$ can query partial key extract, public key extract, private key extract, public key replace, re-encryption key generate, re-encryption oracles, incremental modification, and decrypt oracles. However, adversary is restricted to submit only valid queries to aforementioned oracles.
- Forgery: Finally, the adversary has to output a ciphertext (C_A, C) or (C_B, C) which should be different from ciphertexts received from aforementioned oracles. The adversary succeeds if (C_A, C) or (C_B, C) is a valid ciphertext. If the outcome of the Decrypt oracle is not an error message, adversary $\mathcal{A}dv_1$ succeeds in the game, and $\mathcal{A}dv_1$ has not queried both partial key extract oracle and replace public key oracle or secret key extract oracle of the receiver at any instant of the game.

Definition 2. An incremental certificateless proxy re-encryption scheme $\left(t, q_{par}, q_{pri}, q_{pub}, q_{pub\_rep}\ q_{enc}, q_{re}, q_{re-enc}, \mu, q_{inc}, n; v\right)$ is CPA-secure against unforgeability if, for any time t, the adversary who makes at most q_{par} queries to partial key extract oracle, q_{pub} queries to public key extract, q_{pri} queries to private key extract, q_{pub_rep} queries to public key replace, q_{enc} queries to encryption oracles q_{re} queries to re-encryption key generate, q_{re-enc} queries to re-encryption oracles, and q_{inc} valid queries to incremental modification oracle (with μ as total blocks on which encryption and incremental modification oracles are called), we have advantage of adversary $\mathcal{A}dv_1$ less than v.

2.2.2 | Security against type II adversary in game-II

The master secret key can be accessed by Type II adversary but is not allowed to replace the public key. It can easily compute partial key of any entity so access to private key extract oracle is not required. The rest of the game and access to oracles is similar to Game-I.

3 | PROPOSED INCREMENTAL CERTIFICATELESS PROXY RE-ENCRYPTION (CL-IPRE) SCHEME

A lightweight certificateless incremental proxy re-encryption scheme (CL-iPRE) has been proposed for secure sharing of EHRs relying on the proxy resident cloud for re-encryption operations. We assumed that the public cloud is fully untrusted and cloud-resident proxy server is semitrusted, ie, it follows the protocol but can gather information to deduce private information. The certificateless cryptography solves key escrow problem in traditional public key cryptography. Any incremental cryptographic algorithm must run faster than re-computing the transformation from scratch. For example, if b denotes the block size, then a single modification to a block should take $poly(b)$ time regardless of the number of blocks or total size of entire EHR. The architecture of proposed CL-iPRE scheme is shown in Figure 1.

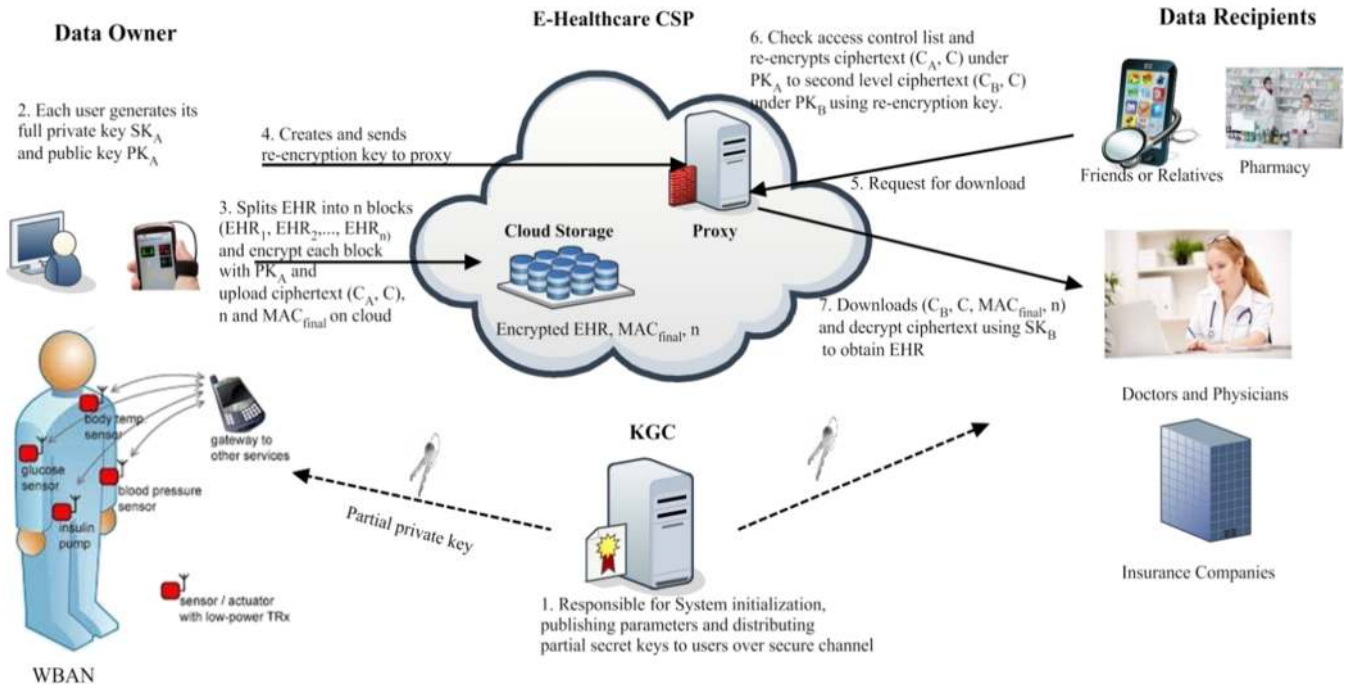

FIGURE 1 Architecture of proposed CL-iPRE scheme for secure sharing of EHRs in MCC

Our proposed incremental CL-iPRE scheme is as follows:

- ***System setup:*** This algorithm is run by a key generation center (KGC). It chooses security parameter k and and elliptic curve E/F_p over prime finite field F_p. Let G be the cyclic subgroup of elliptic curve group with P as generator of order q. It randomly selects master secret key $\in Z_q^*$. It further computes master public key $y = xP$ and announces a list of public parameters $\delta = \{E, F_p, G, P, y\}$ for encryption, decryption, and proxy re-encryption.
- ***Partial key generation:*** This algorithm is executed by KGC, which takes public parameters δ, cloud user identity ID_U, master secret key x, and master public key y as inputs. It randomly chooses $z_U \in Z_q^*$ and computes $W_U = z_U P$ and $h_U = \mathcal{H}_1(ID_U, W_U)$ as U' s partial public key corresponding to user's identity ID_U. It generates $Z_U = (z_U + h_U x)$ as U' s partial private key. It further transmits partial key (Z_U, W_U) to user U.
- ***Set secret value:*** This algorithm is called by every cloud user, which takes as inputs public parameters δ and user's identity ID_U and outputs a secret value $v_U \in Z_q^*$.
- ***Private key generation:*** This algorithm is executed by cloud user U with identity ID_U; accepts public parameters δ, partial private key Z_U, and secret value $v_U \in Z_q^*$ as inputs; and outputs a full secret key $Sk_U = (Z_U - v_U)$ for user U.
- ***Public key generation:*** This algorithm is called by every cloud user U having identity ID_U which computes $V_U = v_U P$ and a public key $Pk_U = Sk_U P$ for user U. Any user who wishes to use this public key can validate the received public key as

$$Pk_U = W_U + h_U y - V_U$$

$$\text{where } h_U = \mathcal{H}_1(ID_U, W_U)$$

- ***Re-encryption key generation***: This algorithm is called by data owner A for generating re-encryption key $Rk_{A\to B} = {}^{Sk_B}/_{Sk_A}$ using secure two-party integer division algorithm,[45] which is forwarded to cloud resident proxy server $\wp$ without leaking any information about Sk_B or Sk_A to data owner, data recipient, or proxy server. This algorithm[45] permits two or more mutually mistrusting parties to evaluate a function on confidential data without revealing further information to the parties involved.
- ***Encryption:*** The mobile data owner A wants to upload data on cloud. Koblitz method[46] is used for encoding a plaintext message as an elliptic curve point EHR. The owner A splits EHR into n blocks $EHR_1, EHR_2, \ldots \ldots, EHR_n$, where each block has fixed size of b bits except possibly the last.

$$EHR = \|_{i=1}^{n} (EHR_i)$$

$$b_j = \left\lceil \frac{s}{n} \right\rceil \text{ where } 1 \le j \le n-1$$

$$b_n = s - \left(\left\lceil \frac{s}{n} \right\rceil * (n-1) \right)$$

where EHR_i denotes the ith block of EHR, s denotes the total size of $EHRs$, b_j denotes the size of the jth block of EHR, and $\left\lceil \frac{s}{n} \right\rceil$ denotes the mathematical floor function to remove the fractional part. To provide confidentiality, data owner A encrypts each block EHR_i before storing on cloud. Data owner A generates a random number $\alpha \in Z_q^*$ and encrypts the message using its public key Pk_A.

$$C_A = \alpha.Pk_A, C_i = EHR_i + \alpha.P$$

$$C = \|_{i=1}^{n} (C_i)$$

where $1 \leq i \leq n$.

Similarly, hash for each block of EHR is generated using the recently proposed SHA-3 algorithm. The hash function has to map b bits of each block to l bits, ie, $h : \{0,1\}^b \rightarrow \{0,1\}^l$ where l is a multiple of 64 bits. The individual hash values are then combined using wordwise addition in the commutative group $((Z_{2^{64}},)^{l/64} \boxplus_{64})$ to generate final hash value for verifying integrity of stored $EHRs$. The group combining operator $\boxplus_{64}$ represents 64-bit wordwise addition of $l/64$ words and $\boxminus_{64}$ represents the inverse of addition operation, ie, wordwise subtraction of $l/64$ words. The final hash value H_{final} is digitally signed to obtain S_{final}, which provides authentication and nonrepudiation.

$$H_i = \mathcal{H}_{\text{SHA}-3} (EHR_i) \text{ where } 1 \leq i \leq n$$

$$H_{\text{final}} = H_1 \boxplus_{64} H_2 \boxplus_{64} \ldots\ldots \boxplus_{64} H_{\text{n}}$$

The encrypted EHRs (C_A, C),the number of blocks n and S_{final} are uploaded on cloud for storage. The data owner needs to save the file name of $EHRs$ and the number of blocks n for each EHR file in its local memory of mobile device.

- ***Proxy re-encryption:*** The data recipient B who wants to download data uploaded by user A requests semitrusted proxy, which alters first-level ciphertext $C_A \in C_1$ as received from data owner A into second-level ciphertext $C_B \in C_2$ for data recipient B. Proxy sends re-encrypted message (C_B, C) to user B for decryption and integrity verification.

$$C_B = C_A Rk_{A \rightarrow B} = \alpha.Sk_A.P \left(\frac{Sk_B}{Sk_A} \right) = \alpha Sk_B P, C_i = EHR_i + \alpha.P$$

$$C = \|_{i=1}^{n} (C_i)$$

where $1 \leq i \leq n$

- ***Decrypt:*** This algorithm is executed by cloud user $U \in \{A, B\}$, which takes ciphertext C_U corresponding to user U as an input and corresponding plain text message is given as output or an error symbol Υ if C_U is invalid. The data owner A decrypts the message using C_A and its secret key Sk_A as follows:

$$EHR_i = C_i - \left(\frac{1}{Sk_A} \right) C_A \text{ where } 1 \leq i \leq n.$$

The data recipient B decrypts the message using C_B and its secret key Sk_B as follows:

$$EHR_i = C_i - \left(\frac{1}{Sk_B} \right) C_B \text{ where } 1 \leq i \leq n.$$

The mobile user concatenates individual blocks of the file to get original EHR and verifies integrity of the downloaded file as follows:

$$EHR = \|_{i=1}^{n}, (EHR_i) \text{ where } 1 \leq i \leq n$$

$$H_i' = \mathcal{H}_{SHA-3}, (EHR_i) \text{ where } 1 \leq i \leq n$$

$$H_{\text{final}}' = H_1 \boxplus_{64} H_2 \boxplus_{64} \ldots\ldots \boxplus_{64} H_n.$$

The data recipient B uses A' s public key to decrypt (de-sign) the digital signature S_{final} and obtains H_{final}. It verifies integrity of EHR received by comparing H_{final} with calculated H_{final}'. If both values match, then integrity is confirmed. EHR is decoded using[46] to obtain the original message.

4 | BLOCK MODIFICATION OPERATIONS

The block modification operations include block(s) update, insertion, and deletion. The overhead involved in dividing $EHRs$ and calculating hash values of individual blocks increase resource utilization on the mobile device as compared to proxy re-encryption schemes; however, incremental

concept improves resource utilization on the mobile device while performing block modification operations on uploaded EHRs as hash values need not to be recalculated from scratch for each block modification query.

4.1 | Incremental block updation operations

Once the hash function is applied on individual blocks and S_{final} is stored on cloud, there is no need to repeat the entire procedure on the mobile device while updating the block(s). We can apply an incremental update operation and perform wordwise-subtraction $\boxminus_{64}$ of old hash values of updated blocks and wordwise-addition $\boxplus_{64}$ of updated hash values of modified blocks. Thus, only two hash operations are required for each modified block. The storage and communication overhead is also reduced because a CSP needs to store the signature value on cloud and the mobile device needs to download the *l*-bit hash value for recomputing the hash value.

Let data owner *A* wants to modify *q* block(s) represented by EHR_{update} in the uploaded record starting from location index *idx*. The data owner *A* needs to download encrypted *EHR*, ie, (C_A, C) and S_{final} corresponding to record *EHR* from cloud storage on its device. C_A is used for encrypting updated blocks as follows:

$$C_{\text{update}_j} = EHR_{\text{update}_j} + \left(\frac{1}{Sk_A}\right) C_A \text{ where } 1 \le j \le q.$$

The data owner *A* then calculates updated hash value H_{new} and sends an update request to CSP along with C_{update}, location index *idx*, number of modified blocks *q*, and H_{new}. The updated hash value H_{new} is digitally signed to obtain S_{new} using the $A's$ private key.

$$H_{\text{update}} = H_{\text{idx}} \boxplus_{64} H_{\text{idx+1}} \boxplus_{64} \ldots\ldots \boxplus_{64} H_{\text{idx+}q}$$

$$H_{\text{new}} = H_{\text{final}} \boxminus_{64} H_{\text{idx}} \boxminus_{64} H_{\text{idx+1}} \boxminus_{64} \ldots\ldots \boxminus_{64} H_{\text{idx+q}} \boxplus_{64} H_{\text{update}}.$$

CSP updates encrypted *EHR* and stores S_{new} obtained by signing H_{new} corresponding to stored *EHR*.

$$C = \|_{i=1}^{\text{idx}} (C_i) \left\| \|_{j=1}^{q} C_{\text{update}_j} \right\|_{k=q+1}^{n} (C_k)$$

4.2 | Incremental block insertion operations

The mobile data owner *A* who wishes to insert *q* new blocks represented by EHR_{insert} after location index *idx* in previously uploaded record *EHR* sends a download request to the cloud server. The data owner *A* downloads (C_A, C) and S_{final} corresponding to record EHR. C_A is used for encrypting new blocks as follows:

$$C_{\text{insert}_i} = EHR_{\text{insert}_i} + \left(\frac{1}{Sk_A}\right) C_A \text{ where } 1 \le i \le q$$

The data owner *A* downloads S_{final} from cloud corresponding to stored *EHR* and apply $\boxplus_{64}$ to obtain hash values of newly inserted blocks. Thereafter, it combines it to H_{insert} to obtain new hash value denoted by H_{new}. The data owner then sends an insert request to CSP including S_{new} obtained by signing H_{new}, location *loc* after which blocks are to be inserted and newly encrypted blocks C_{insert_i} where $1 \le i \le q$.

$$H_{\text{insert}} = H_{\text{insert}_1} \boxplus_{64} H_{\text{insert}_2} \boxplus_{64} \ldots\ldots \boxplus_{64} H_{\text{insert}_q}$$

$$H_{\text{new}} = H_{\text{final}} \boxplus_{64} H_{\text{insert}}.$$

CSP insert(s) the block(s) into corresponding encrypted *EHR* and updates ciphertext as follows:

$$C = \|_{i=1}^{\text{idx}} (C_i) \left\| \|_{j=1}^{q} (C_j) \right\|_{k=\text{idx+1}}^{n} (C_k)$$

S_{final} value is updated to S_{new} and number of blocks $b = b + q$ for corresponding *EHR*. The block insertion can be done at multiple locations also.

4.3 | Incremental block deletion operations

To delete *q* blocks at some random location index *idx* from uploaded *EHR* on cloud, data owner *A* downloads S_{final} of corresponding *EHR* from the cloud storage. The data owner recalculates hash as follows:

$$H_{\text{new}} = H_{\text{final}} \boxminus_{64} H_{\text{idx}} \boxminus_{64} H_{\text{idx+1}} \boxminus_{64} \ldots\ldots \boxminus_{64} H_{\text{idx+}q}$$

where H_{idx}, $H_{idx+1}, \ldots \ldots, H_q$ is the corresponding hash values of deleted block(s). The data owner sends a delete request to the CSP along with recalculated value S_{new}, location index *idx*, and number of blocks to be deleted *q*. The CSP deletes corresponding block(s) in stored encrypted *EHR* and stores updated signature corresponding to stored *EHR*.

$$C = \left\| _{i=1}^{idx-1} (C_i) \right\|_{j=idx+q+1}^{n} (C_j)$$

5 | FORMAL ANALYSIS AND VERIFICATION

The basic ECC encryption algorithm is semantically secure relying on elliptic-curve discrete logarithm problem, ie, given a base point, it is infeasible to find discrete logarithm of an elliptic curve. Since our proposed CL-iPRE scheme is based on well-recognized atomic BBS re-encryption whose security is proven in the work of Blaze et al[12] and indistinguishablity and unforgeability of incremental encryption is well proved in the work of Buonanno et al,[32] we will bound our analysis to the verification of the proposed CL-iPRE scheme using formal methods as used by Khan et al.[23,36] Figure 2 represents the HLPN model of the proposed scheme. The data types and mappings are represented in Tables 1 and 2. In the HLPN model, the circle represents places $p \in \mathfrak{P}$, where $\mathfrak{P} = \{KGC, Proxy, Mobile\ User, Cloud\}$, and the rectangular blocks represent transition $t \in T$.

The system starts with setup and partial private key generation phase for each user having access to data partition *d* on cloud using transition *Gen _ Partial _ Key* shown by the following rule:

$$R(Gen\_Partial\_Key) = \forall x_1 \in X_1, \forall x_2 \in X_2 \mid$$

$$x_2[3] \leftarrow gen\_z_i(x_1[2]) \wedge x_2[4] \leftarrow gen\_x(x_1[1]) \wedge x_2[1] \leftarrow gen\_\delta(x_1[1]) \wedge$$

$$x_2[5] \leftarrow gen\_Z_i(x_1[2], x_2[3], x_2[4])$$

$$X_2' = X_2 \cup \{x_2[1], x_2[3], x_2[4], x_2[5]\}.$$

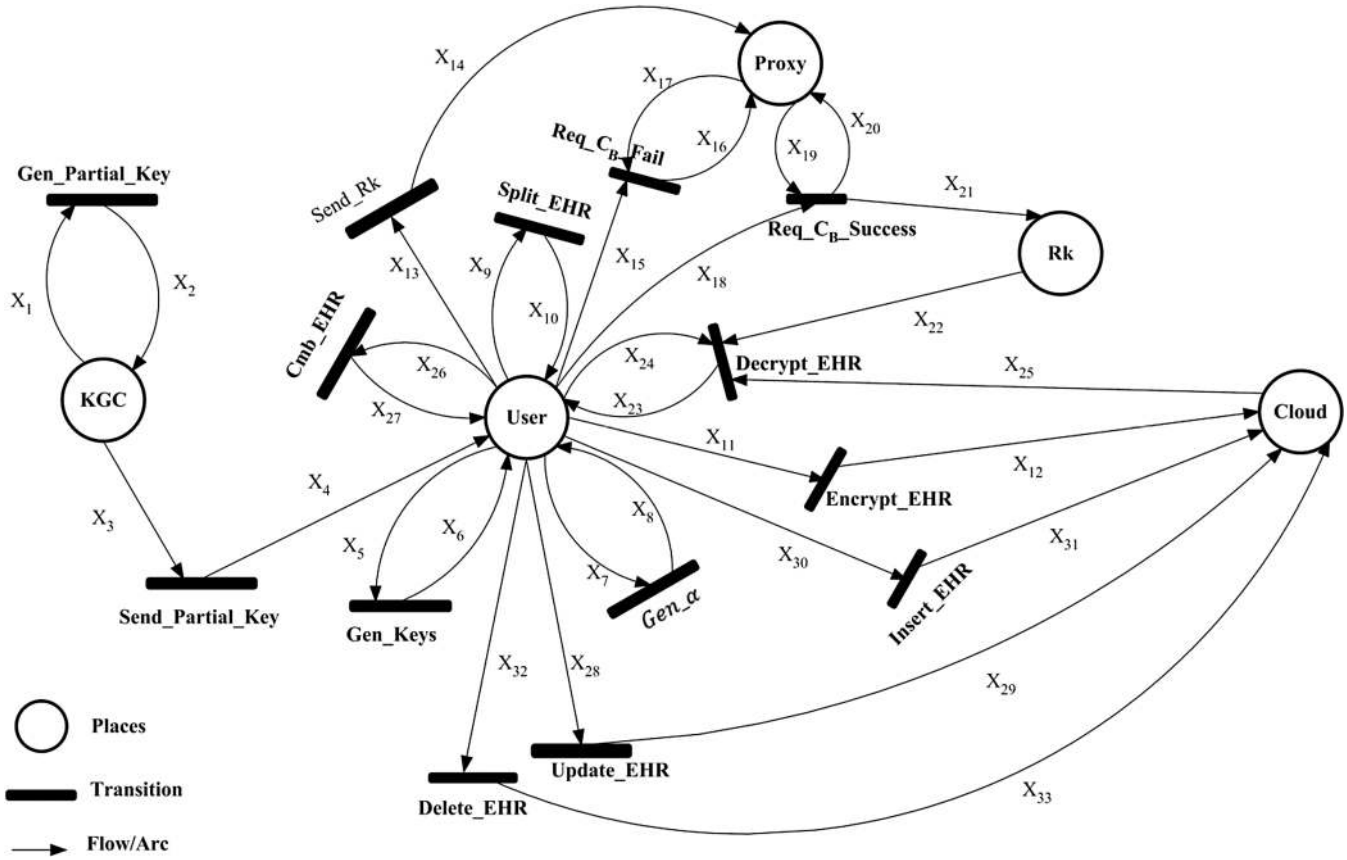

FIGURE 2 HLPN model of proposed CL-iPRE scheme

TABLE 1 Data types for CL-iPRE scheme

Data type	Meaning
ID_i	Unique identity identifier
D_i	Unique identifier of data partition *i* over cloud
α	A random number
δ	System parameters generated by KGC
Sk_i	Secret key of user *i*
Pk_i	Public key of user *i*
Z_i	Partial private key of user *i*
W_i	Partial public key of user *i*
EHR	A record to be encrypted and stored on cloud
n	Total number of blocks in *EHR*
EHR_i	String representing i^{th} block of *EHR*
C_i	Encrypted block EHR_i
C	Encrypted *EHR* such that $C = \|_{i=1}^{n} (C_i)$
C_A	Number representing α. Pk_A
C_B	Number representing α. Pk_B
H_i	Hash value of EHR_i
H_{final}	Final hash value of n blocks
C_{update}	A record to be encrypted and stored on cloud
EHR_{update}	Blocks user wants to update
C_{update}	Encrypted *EHR* of updated blocks
H_{new}	Hash value after modification operation
q	Number of blocks to be modified
$Rk_{A \to B}$	Re-encryption key from mobile user *A* to *B*
EHR_{insert}	New blocks user wants to insert in uploaded file
C_{insert}	Encrypted *EHR* of newly inserted blocks

TABLE 2 Places and mappings used in HLPN model of CL-iPRE

Place	Mapping
$\varphi(KGC)$	$\mathfrak{P} = (\delta \times \text{ID}_i \times z_i \times Z_i \times x \times W_i)$
$\varphi(Proxy)$	$\mathfrak{P} = (\delta \times ID_i \times C_A \times C_B \times Rk_{A \to B})$
$\varphi(Mobile\_User)$	$\mathfrak{P} = (ID_i \times v_i \times Z_i \times W_i \times Sk_i \times Pk_i \times \delta \times \alpha \times EHR \times n \times EHR_i \times C \times C_i \times H_{\text{final}} \times Rk_{A \to B} \times H_i \times q \times C_{\text{update}} \times EHR_{\text{update}} \times H_{\text{new}} \times C_{\text{insert}} \times EHR_{\text{insert}})$
$\varphi(Cloud)$	$\mathfrak{P} = (D_i \times ID_i \times C_A \times C \times H_{\text{final}})$
$\varphi(Rk)$	$\mathfrak{P} = (C_B)$

The generated partial keys are forwarded to authorized cloud users by transition *Send_Partial_Key* by the following rule:

$$R(Send\_\text{Partial\_Key}) = \forall x_3 \in X_3, \forall x_4 \in X_4 \mid$$
$$x_4[3] \leftarrow (x_3[3]) \wedge x_4[6] \leftarrow (x_3[1])$$
$$X_4' = X_4 \cup \{x_4[3], x_4[6]\}.$$

The user *i* generates secret key Sk_i and public key Pk_i through transition *Gen_Keys* by the following rule:

$$R(Gen\_Keys) = \forall x_5 \in X_5, \forall x_6 \in X_6 \mid$$
$$x_6[2] \leftarrow gen\_v_i(x_5[1]) \wedge x_6[4] \leftarrow gen\_Sk_i(x_6[2], x_5[3]) \wedge x_6[5] \leftarrow gen\_\text{Pk}_i(x_5[1]) \wedge$$
$$X_6' = X_6 \cup \{x_6[2], x_6[4], x_6[5]\}.$$

The parameter α is generated by cloud user through transition *Gen_*α and the following rule:

$$R(Gen\_\alpha) = \forall x_7 \in X_7, \forall x_8 \in X_8 \mid$$
$$x_8[7] \leftarrow gen\_\alpha(x_7[1]) \wedge$$
$$X_8' = X_8 \cup \{x_8[7]\}.$$

The data owner who wants to upload encrypted EHR on cloud splits it into multiple blocks by following rule over transition *Split_EHR*:

$$R\left(Split\_EHR\right) = \forall x_9 \in X_9, \forall x_{10} \in X_{10} \mid$$
$$x_{10}\left[10\right] \leftarrow split\left(x_9\left[8\right], x_9\left[9\right]\right) \wedge$$
$$X'_{10} = X_{10} \cup \left\{x_{10}\left[10\right]\right\}.$$

The mobile data owner encrypts *EHR* and generates corresponding *MAC* represented by transaction *Encrypt_EHR* and the following rule:

$$R\left(Encrypt\_EHR\right) = \forall x_{11} \in X_{11}, \forall x_{12} \in X_{12} \mid$$
$$x_{12}\left[1\right] \leftarrow x_{11}\left[1\right] \wedge x_{12}\left[12\right] \leftarrow encrypt\left(x_{11}\left[10\right], x_{11}\left[9\right]\right) \wedge x_{12}\left[11\right] \leftarrow concat\left(x_{12}\left[12\right]\right) \wedge$$
$$x_{12}\left[13\right] \leftarrow concat\left(gen\_mac\left(x_{12}\left[10\right]\right)\right) \wedge$$
$$X'_{11} = X_{11} \cup \left\{x_{12}\left[1\right], x_{12}\left[11\right], x_{12}\left[12\right], x_{12}\left[13\right]\right\}.$$

The mobile data owner sends re-encryption key $Rk_{A \rightarrow B}$ to proxy.

$$R\left(Sends\_Rk\right) = \forall x_{13} \in X_{13}, \forall x_{14} \in X_{14} \mid$$
$$x_{14}\left[5\right] \leftarrow x_{13}\left[14\right] \wedge$$
$$X'_{14} = X_{14} \cup \left\{x_{14}\left[5\right]\right\}.$$

The mobile user *B* who wishes to view *EHR* uploaded by *A* request proxy for download. The proxy checks its ACL, and if an entry for user *B* is missing, the request fails as shown by the following rule:

$$R\left(Req\_C_B\_fails\right) = \forall x_{15} \in X_{15}, \forall x_{16} \in X_{16}, \forall x_{17} \in X_{17} \mid$$
$$x_{16}\left[2\right] \leftarrow x_{15}\left[1\right] \wedge$$
$$X'_{16} = X_{16} \wedge$$
$$X'_{17} = X_{17}.$$

If the requesting user *B* has access rights for *EHR* uploaded by *A*, proxy re-encrypts C_A to C_B using re-encryption key $Rk_{A \rightarrow B}$ received from *A* and sends C_B to user *B* as represented by transition *Req_C_B_success*:

$$R\left(Req\_C_B\_success\right) = \forall x_{18} \in X_{18}, \forall x_{19} \in X_{19}, \forall x_{20} \in X_{20}, \forall x_{21} \in X_{21} \mid$$
$$x_{19}\left[2\right] \leftarrow x_{18}\left[1\right] \wedge x_{20}\left[4\right] \leftarrow prod\left(x_{19}\left[3\right], x_{19}\left[5\right]\right)$$
$$X'_{19} = X_{19} \wedge X'_{21} = X_{21} \wedge$$
$$\forall x_{22} \in X_{22} \mid$$
$$x_{21}\left[1\right] \leftarrow x_{20}\left[4\right] \wedge x_{22}\left[1\right] \leftarrow x_{21}\left[1\right]$$
$$X'_{22} = X_{22} \cup \left\{x_{22}\left[1\right]\right\}.$$

The mobile user *B* decrypts encrypted *EHR* (C_B, C) into plaintext *EHR* at transition *Decrpyt_*C_B with the following rule:

$$R\left(Decrypt\_C_B\right) = \forall x_{22} \in X_{22}, \forall x_{23} \in X_{23}, \forall x_{24} \in X_{24}, \forall x_{25} \in X_{25} \mid$$
$$x_{23}\left[10\right] \leftarrow decrypt\left(x_{25}\left[4\right], x_{22}\left[1\right], x_{23}\left[1\right]\right) \wedge$$
$$X'_{23} = X_{23} \cup \left\{x_{23}\left[10\right]\right\}.$$

After decryption, mobile user *B* concatenates the blocks and verifies integrity with transition *Cmb_EHR* and the following rule:

$$R\left(Cmb\_EHR\right) = \forall x_{26} \in X_{26}, \forall x_{27} \in X_{27} \mid$$
$$x_{27}\left[15\right] \leftarrow gen\_mac\left(x_{26}\left[10\right]\right) \wedge x_{27}\left[13\right] \leftarrow gen\_mac\_final\left(x_{27}\left[15\right]\right) \wedge$$
$$x_{27}\left[8\right] \leftarrow x_{26}\left[10\right]$$
$$X'_{27} = X_{27} \cup \left\{x_{27}\left[15\right], x_{27}\left[13\right], x_{27}\left[8\right]\right\}.$$

When data owner A wishes to modify the *EHR* stored on cloud, it calls *Update_EHR*, *Insert_EHR*, and *Delete_EHR* transitions:

$$R\left(Update\_EHR\right) = \forall x_{28} \in X_{28}, \forall x_{29} \in X_{29} \mid$$
$$x_{28}\left[17\right] \leftarrow encrypt\left(x_{28}\left[18\right], x_{28}\left[7\right], x_{28}\left[16\right]\right) \wedge$$
$$x_{28}\left[19\right] \leftarrow gen\_mac\_final\left(x_{28}\left[13\right], x_{28}\left[18\right], x_{28}\left[16\right]\right) \wedge$$
$$x_{28}\left[11\right] \leftarrow update\left(x_{28}\left[17\right]\right) \wedge x_{28}\left[13\right] \leftarrow update\left(x_{28}\left[19\right]\right) \wedge$$
$$x_{29}\left[4\right] \leftarrow x_{28}\left[11\right] \wedge x_{29}\left[5\right] \leftarrow x_{28}\left[13\right] \wedge$$
$$X'_{28} = X_{28} \cup \left\{x_{28}\left[17\right], x_{28}\left[11\right], x_{28}\left[13\right], x_{28}\left[19\right]\right\}$$
$$X'_{29} = X_{29} \cup \left\{x_{29}\left[4\right], x_{29}\left[5\right]\right\}$$
$$R\left(Insert\_EHR\right) = \forall x_{30} \in X_{30}, \forall x_{31} \in X_{31} \mid$$
$$x_{30}\left[20\right] \leftarrow encrypt\left(x_{30}\left[21\right], x_{30}\left[7\right], x_{30}\left[16\right]\right) \wedge$$
$$x_{30}\left[19\right] \leftarrow gen\_mac\_final\left(x_{30}\left[13\right], x_{30}\left[21\right], x_{30}\left[16\right]\right) \wedge$$
$$x_{30}\left[11\right] \leftarrow insert\left(x_{30}\left[20\right]\right) \wedge x_{30}\left[13\right] \leftarrow insert\left(x_{30}\left[19\right]\right) \wedge$$
$$x_{31}\left[4\right] \leftarrow x_{30}\left[11\right] \wedge x_{31}\left[5\right] \leftarrow x_{30}\left[13\right] \wedge$$
$$X'_{30} = X_{30} \cup \left\{x_{30}\left[20\right], x_{30}\left[11\right], x_{30}\left[13\right], x_{30}\left[19\right]\right\}$$
$$X'_{31} = X_{31} \cup \left\{x_{31}\left[4\right], x_{31}\left[5\right]\right\}$$
$$R\left(Delete\_EHR\right) = \forall x_{32} \in X_{32}, \forall x_{33} \in X_{33} \mid$$
$$x_{32}\left[11\right] \leftarrow delete\left(x_{32}\left[16\right]\right) \wedge x_{32}\left[19\right] \leftarrow gen\_mac\_final\left(x_{32}\left[16\right], x_{32}\left[13\right]\right) \wedge$$
$$x_{33}\left[4\right] \leftarrow x_{32}\left[11\right] \wedge x_{33}\left[5\right] \leftarrow x_{32}\left[19\right] \wedge$$
$$X'_{32} = X_{32} \cup \left\{x_{32}\left[11\right], x_{32}\left[19\right]\right\}$$
$$X'_{33} = X_{33} \cup \left\{x_{33}\left[4\right], x_{33}\left[5\right]\right\}.$$

The aforementioned HLPN model was translated to SMT-Lib and Z_3 solver is used for verification of the proposed CL-iPRE scheme to check if it works according to specifications and gives accurate results. The solver verifies that the encryption of *EHR* is done correctly by mobile data owner *A* as specified by the system. Decryption requests for uploaded *EHR* are handled by the proxy correctly, and data recipient *B* is able to decrypt data correctly from re-encrypted ciphertext. The block modification operations are also verified if the mobile user is able to get modified content correctly.

6 | RESULTS AND COMPARATIVE ANALYSIS

The proposed scheme is evaluated in terms of turnaround time and energy consumption on the mobile device for block updation operations in comparison to proxy re-encryption scheme (PReS) and I-PReS.[36] Khan et al[36] assumed proxy as fully trusted entity possessing secret keys that can decrypt message of any entity suffering from key escrow problem. It involves expensive bilinear pairing operations and downloading of intermediate MAC values of all blocks from cloud by mobile user for block modification operations thereby, increasing communication, computation, and storage overhead on mobile device as well as on cloud. Android SDK is used for developing mobile cloud client application deployed on Nexus smartphone with 1-GB RAM and 2.53-GHz processor. A web instance is hosted on GAE, which provides cloud-based infrastructure for testing Android apps. A standard cryptographic library, MIRACL,[47] has been used to compute the running time of the CL-iPRE scheme. To achieve 1024-bit RSA-level security for bilinear pairing–based schemes, we have used Type A pairing defined over the elliptic curve $E/F_p : y^2 = x^3+x$ with embedding degree 2, where q is a 160-bit Solinas prime and p is a 512-bit prime. To achieve an equivalent security in the ECC-based scheme, we employed the verifiably random elliptic curve secp160r1[48] over F_p, where $p = 2^{160} - 2^{31} - 1$. We have used SHA3-512 in the proposed CL-iPRE scheme as hash function as it is more resistant to pre-imaging and collision attacks than its predecessors. SHA-2 was used in I-PReS. Each experiment is repeated five times and average results are taken. We have evaluated turnaround time and energy consumption for files of different sizes as shown in Table 3 with eight blocks in each file. We have taken readings while updating one, two, and four blocks represented as CL-iPRE(1), CL-iPRE(2), and CL-iPRE(4), respectively.

$$Total\ turnaround\ time = t_{\text{read}} + t_{\text{encrypt}} + t_{\text{upload}}$$

where *t*read refers to file reading time, t_{encrypt} represents block updating time and t_{upload} represents block uploading time.

$$Energy\ consumed = E_{\text{read}} + E_{\text{encrypt}} + E_{\text{upload}},$$

TABLE 3 Data set used for file modification operations

File size in bytes	Total files	Block size in bytes
51 200	50	6400
102 400	50	12 800
153 600	50	19 200
204 800	50	25 600

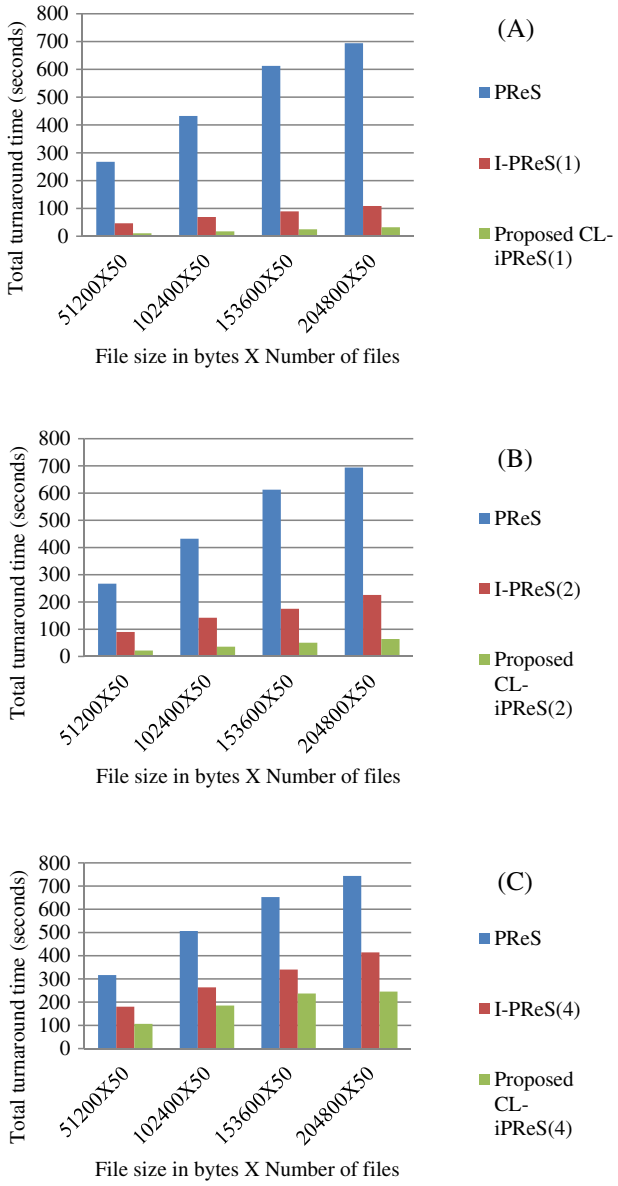

FIGURE 3 Turnaround time while block updating operations on data set. A, Updating one block; B, Updating two blocks; C Updating four blocks

where E_{read} refers to energy consumed in file reading, $E_{encrypt}$ represents energy consumed in block updating, and E_{upload} represents energy consumed in file uploading.

Figures 3 and 4 show the variation in turnaround time (in seconds) and energy consumption (in %) with different file sizes. As depicted in Figure 3, the average turnaround time (with files of varying sizes) in the proposed CL-iPRE scheme is 26.49% of I-PReS and 4.1% of PReS with one modified block, 26.52% of I-PReS and 8.4% of PReS with two modified blocks, and 33.96% of I-PReS and 20.43% of PReS with four modified blocks. The average energy consumption (with files of varying sizes) in the proposed CL-iPRE scheme is 67.22% of I-PReS and 11.03% of PReS with one modified block, 71.37% of I-PReS and 23.78% of PReS with two modified blocks, and 78.63% of I-PReS and 48.17% of PReS with four modified blocks as shown in Figure 4.

This clearly depicts that the proposed scheme is computationally efficient than existing schemes PReS and I-PReS for file modification operations. The communication overhead of proposed CL-iPRE scheme is less as compared to I-PReS because individual MAC values need not to be uploaded on cloud for storage and cost of elliptic curve point multiplication is less as compared to pairing based scalar multiplication.

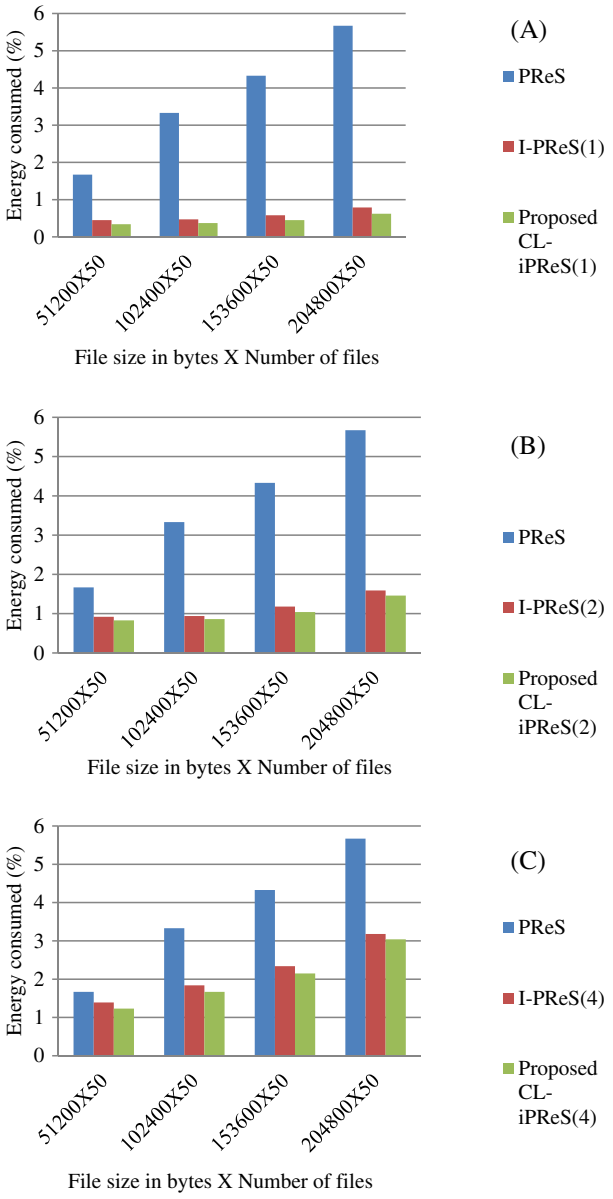

FIGURE 4 Energy consumption while block updating operations on data set. A, Updating one block; B, Updating two blocks; C, Updating four blocks

7 | CONCLUSION AND FUTURE SCOPE

Secure EHRs storage and sharing is a serious issue while utilizing underlying cloud infrastructure. Certificate-based cryptography incurs huge overhead involved in management of certificates, and identity-based cryptography suffers from key-escrow problem. Certificateless incremental proxy re-encryption primitive can overpower these issues and offer secure EHR sharing. Recently, the proposed I-PRes scheme involves expensive bilinear pairing operations and downloading of intermediate MAC values of all blocks from cloud by mobile user while block insertion, deletion, or update operation, thereby increasing communication, computation, and storage overhead on a mobile device as well as on the cloud. A lightweight pairing-free CL-iPRE scheme for secure sharing of EHRs in mobile cloud computing paradigm has been proposed based on elliptic curves. The proposed scheme handles block modification operations efficiently and reduces storage overhead on mobile device and public cloud making it feasible for mobile devices. In future, we can further design a group-based incremental proxy re-encryption scheme.

ORCID

Tarunpreet Bhatia https://orcid.org/0000-0003-0434-2799

REFERENCES

1. Atallah MJ, Pantazopoulos KN, Rice JR, Spafford EE. Secure outsourcing of scientific computations. *Adv Comput*. 2002;54:215-272.
2. Guo P, Wang J, Ji S, Geng XH, Xiong NN. A lightweight encryption scheme combined with trust management for privacy-preserving in body sensor networks. *J Med Syst*. 2015;39(12):190.
3. Chen X, Li J, Susilo W. Efficient fair conditional payments for outsourcing computations. *IEEE Trans Inf Forensics Secur*. 2012;7(6):1687-1694.

4. Backes M, Fiore D, Reischuk RM. Verifiable delegation of computation on outsourced data. In: Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security; 2013; Berlin, Germany.
5. Chen X, Li J, Ma J, Tang Q, Lou W. New algorithms for secure outsourcing of modular exponentiations. *IEEE Trans Parallel Distributed Syst*. 2014;25(9):2386-2396.
6. Bhatia T, Verma AK. Data security in mobile cloud computing paradigm: a survey, taxonomy and open research issues. *J Supercomput*. 2017;73(6):2558-2631.
7. Muhammed T, Mehmood R, Albeshri A, Katib I. UbeHealth: a personalized ubiquitous cloud and edge-enabled networked healthcare system for smart cities. *IEEE Access*. 2018;6:32258-32285.
8. Nidhya R, Karthik S. Security and privacy issues in remote healthcare systems using wireless body area networks. In: Maheswar R, Kanagachidambaresan G, Jayaparvathy R, Thampi S, eds. *Body Area Network Challenges and Solutions*. Cham, Switzerland: Springer: 2019, 37-53. *EAI/Springer Innovations in Communication and Computing*.
9. Turner V, Gantz JF, Reinsel D, Minton S. The digital universe of opportunities: Rich data and the increasing value of the internet of things. IDC Analyze the Future. 2014.
10. Cisco. Cisco VNI: Forecast and Methodology, 2015–2020 White Paper. 2016. http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html. Accessed March 23, 2017.
11. Cisco. Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2016–2021 White Paper. March 2017. http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html. Accessed March 23, 2017.
12. Blaze M, Bleumer G, Strauss M. Divertible protocols and atomic proxy cryptography. In: Nyberg K, ed. *Advances in Cryptology—EUROCRYPT'98*. Berlin, Germany: Springer; 1998:127-144.
13. Elgamal T. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans Inf Theory*. 1985;31(4):469-472.
14. Ren W, Yu L, Gao R, Xiong F. Lightweight and compromise resilient storage outsourcing with distributed secure accessibility in mobile cloud computing. *Tsinghua Sci Technol*. 2011;16(5):520-528.
15. Hsueh S-C, Lin J-Y, Lin M-Y. Secure cloud storage for convenient data archive of smart phones. Paper presented at: 2011 IEEE 15th International Symposium on Consumer Electronics (ISCE); 2011; Singapore. https://doi.org/10.1109/ISCE.2011.5973804
16. Khan AN, Kiah MM, Ali M, Madani SA, Shamshirband S. BSS: block-based sharing scheme for secure data storage services in mobile cloud environment. *J Supercomput*. 2014;70(2):946-976. https://doi.org/10.1007/s11227-014-1269-8
17. Jia W, Zhu H, Cao Z, Wei L, Lin X. SDSM: a secure data service mechanism in mobile cloud computing. Paper presented at: 2011 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS); 2011; Shangai, China. https://doi.org/10.1109/INFCOMW.2011.5928784
18. Zhou Z, Huang D. Efficient and secure data storage operations for mobile cloud computing. Paper presented at: 2012 8th International Conference on Network and Service Management (CNSM) and 2012 Workshop on Systems Virtualiztion Management (SVM); Las Vegas, NV; 2012.
19. Yang J, Wang H, Wang J, Tan C, Yu D. Provable data possession of resource-constrained mobile devices in cloud computing. *J Networks*. 2011;6(7):1033-1040.
20. Tysowski PK, Hasan MA. Re-encryption-based key management towards secure and scalable mobile applications in clouds. *IACR Cryptol EPrint Arch*. 2011;2011:668.
21. Khan AN, Kiah MM, Ali M, Shamshirband S. A cloud-manager-based re-encryption scheme for mobile users in cloud environment: a hybrid approach. *J Grid Comput*. 2015;13(4):651-675.
22. Mollah MB, Azad MA, Vasilakos A. Secure data sharing and searching at the edge of cloud-assisted internet of things. *IEEE Cloud Comput*. 2017;4(1):34-42.
23. Khan AN, Ali M, Khan AU, et al. A comparative study and workload distribution model for re-encryption schemes in a mobile cloud computing environment. *Int J Commun Syst*. 2017;30(16):e3308.
24. Zhang Y, Zheng D, Deng RH. Security and privacy in smart health: efficient policy-hiding attribute-based access control. *IEEE Internet Things J*. 2018;5(3):2130-2145.
25. Zhang Y, Chen X, Li J, Wong DS, Li H. Anonymous attribute-based encryption supporting efficient decryption test. In: Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security; Hangzhou, China; 2013.
26. Lai J, Deng RH, Li Y. Expressive CP-ABE with partially hidden access structures. In: Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security; 2012; Seoul, Korea.
27. Ali M, Abbas A, Khan U, Khan SU. SeSPHR: a methodology for secure sharing of personal health records in the cloud. *IEEE Trans Cloud Comput*. 2018. https://doi.org/10.1109/TCC.2018.2854790
28. Bhatia T, Verma AK, Verma G. Secure sharing of mobile personal healthcare records using certificateless proxy re-encryption in cloud. *Trans Emerg Telecommun Technol*. 2018;29(6):e3309.
29. Modi KJ, Kapadia N. Securing healthcare information over cloud using hybrid approach. In: Panigrahi C, Pujari A, Misra S, Pati B, Li KC, eds. *Progress in Advanced Computing and Intelligent Engineering*. Singapore: Springer; 2019:63-74. *Advances in Intelligent Systems and Computing*.
30. Bellare M, Goldreich O, Goldwasser S. Incremental cryptography: the case of hashing and signing. In: Desmedt YG, ed. *Advances in Cryptology — CRYPTO'94. CRYPTO 1994*. Berlin, Germany: Springer; 1994:216-233. Lecture Notes in Computer Science; vol. 839.
31. Bellare M, Goldreich O, Goldwasser S. Incremental cryptography and application to virus protection. In: Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing; 1995; Las Vegas, NV.
32. Buonanno E, Katz J, Yung M. Incremental unforgeable encryption. In: Matsui M, ed. *Fast Software Encryption*. Berlin, Germany: Springer; 2001:109-124. *Lecture Notes in Computer Science*; vol. 2355.
33. Mironov I, Pandey O, Reingold O, Segev G. Incremental deterministic public-key encryption. In: Proceedings of the 31st Annual international conference on Theory and Applications of Cryptographic Techniques; 2012; Cambridge, UK.
34. Khan AN, Kiah MM, Khan SU, Madani SA, Khan AR. A study of incremental cryptography for security schemes in mobile cloud computing environments. Paper presented at: 2013 IEEE Symposium on Wireless Technology and Applications (ISWTA); 2013; Kuching, Malaysia.

35. Itani W, Kayssi A, Chehab A. Efficient healthcare integrity assurance in the cloud with incremental cryptography and trusted computing. In: Moumtzoglou A, Kastania A, eds. *Cloud Computing Applications for Quality Health Care Delivery*. Hershey, PA: IGI Global; 2014; 102-115.
36. Khan AN, Kiah MM, Madani SA, Ali M, Shamshirband S. Incremental proxy re-encryption scheme for mobile cloud computing environment. *J Supercomput*. 2014;68(2):624-651.
37. Secure Hash Standard. FIPS PUB 180-1. Public Law 100-235. Gaithersburg, MD: National Institute of Standards and Technology, U.S. Department of Commerce; 1995.
38. Secure Hash Standard. FIPS PUB 180-2. Gaithersburg, MD: National Institute of Standards and Technology, U.S. Department of Commerce; 2004.
39. Dworkin MJ. SHA-3 Standard Permutation-Based Hash and Extendable-Output Functions (DRAFT FIPS PUB 202). http://csrc.nist.gov/publications/PubsDrafts.html#FIPS-202. Accessed March 25, 2017.
40. Merkle RC. A digital signature based on a conventional encryption function. In: *Advances in Cryptology — CRYPTO '87*. Berlin, Germany: Springer: 2006:369-378.
41. Bertoni G, Daemen J, Peeters M, Van Assche G. Sufficient conditions for sound tree and sequential hashing modes. *Int J Inf Secur*. 2014;13(4):335-353.
42. Wang X, Yao AC, Yao F. Cryptanalysis on SHA-1. Paper presented at: Cryptographic Hash Workshop hosted by NIST; 2005; Gaithersburg, MD.
43. CWI. Google announce first collision for Industry Security Standard SHA-1. Accessed February 23, 2017.
44. Mihajloska H, Gligoroski D, Samardjiska S. Reviving the idea of incremental cryptography for the zettabyte era use case: incremental hash functions based on SHA-3. Paper presented at: International Workshop on Open Problems in Network Security; 2015; Zurich, Switzerland.
45. Dahl M, Ning C, Toft T. On secure two-party integer division. Paper presented at: International Conference on Financial Cryptography and Data Security; 2012; Bonaire, The Netherlands.
46. Koblitz N. Elliptic curve cryptosystems. *Math Comput*. 1987;48(177):203-209.
47. Shamus Software Ltd. MIRACL library. https://certivox.org/display/EXT/MIRACL. Accessed September 20, 2016.
48. The Certicom Corporation. SEC 2: Recommended Elliptic Curve Domain Parameters. http://www.secg.org/SEC2-Ver-1.0.pdf. Accessed September 20, 2017.

How to cite this article: Bhatia T, Verma AK, Sharma G. Towards a secure incremental proxy re-encryption for e-healthcare data sharing in mobile cloud computing. *Concurrency Computat Pract Exper*. 2019;e5520. https://doi.org/10.1002/cpe.5520