HealthFort: A Cloud-Based eHealth System With Conditional Forward Transparency and Secure Provenance via Blockchain

Shiyu Li, Yuan Zhang, *Member, IEEE*, Chunxiang Xu, *Member, IEEE*, Nan Cheng, *Member, IEEE*, Zhi Liu, *Member, IEEE*, Yicong Du, and Xuemin Shen, *Fellow, IEEE*

Abstract—In this paper, we propose a servers-aided password-based subsequent-key-locked encryption mechanism to ensure the confidentiality of outsourced electronic health records (EHRs). The encryption mechanism achieves conditional forward transparency: a doctor can only access a patient's EHRs related to the current diagnosis with the patient's delegation. It also achieves portability: to delegate a doctor for accessing a specific part of EHRs, the patient only needs to send one key (at most 256 bits) in addition to the delegation information to the doctor; the patient does not need to maintain any secret in a local device. Then, we propose a blockchain-based secure EHR provenance mechanism, where a data structure of EHR provenance record is designed to precisely reflect the EHRs' provenance information; a smart contract on a public blockchain is deployed to secure both EHRs and the corresponding provenance records. Finally, we develop a cloud-based eHealth system, dubbed HealthFort, based on the two mechanisms. Security analysis and comprehensive performance evaluation are conducted to demonstrate that HealthFort is secure and efficient.

Index Terms—Cloud-based eHealth system, EHR confidentiality, secure data provenance, blockchain

1 Introduction

As modern eHealth systems are data-intensive, outsourcing EHRs to a cloud server is practical and popular. The wide deployment of cloud storage services has brought great benefits in managing electronic health records (EHRs). For instance, patients' EHRs could be well maintained and accessed in a flexible and convenient way [2], [3], [4], [5]. Actually, the recent outbreak of COVID-19 has made the benefits more valuable than ever and demonstrated that cloud-based eHealth systems play a vital role in fighting against the virus [6].

Despite the appealing advantages of the cloud-based eHealth systems, critical threats towards the outsourced EHRs have been raised. Specifically, EHRs are the most sensitive data for patients, and both external adversaries (hackers) and internal adversaries (a compromised cloud server) may always try to retrieve the contents of outsourced EHRs for profits [7], [8], [9]. Moreover, the outsourced EHRs are confronted with deletion and modification in many cases [10], [11]. Typically, when a medical malpractice occurs, the hospital and doctor may collude with the cloud server to modify or delete a target patient's outsourced EHRs for the avoidance of responsibility.

To ensure the security of outsourced EHRs, two key techniques, encryption [12], [13], [14], [15], [16] and data provenance [17], [18], [19], [20], [21], [22], [23], could be utilized, where the former guarantees EHRs' confidentiality so as to prevent the outsourced EHRs from leakage, and the latter keeps track of what happens to the outsourced EHRs throughout their lifecycle. Whereas, directly deploying existing encryption techniques (including both symmetric-key ones and public-key ones) cannot meet the functionality requirements of actual cloud-based eHealth systems. Particularly, in a consultation, a doctor needs to access the patient's EHRs for diagnosing. However, for privacy protection, the doctor should only be able to access the EHRs that are related to the current diagnosis. Such a requirement is

- *Shiyu Li and Yicong Du are with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, Sichuan 610056, China. E-mail: Shai_Li@yeah.net, xlwmfh@163.com.*
- *Yuan Zhang is with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, Sichuan 610056, China, and also with the Fujian Key Laboratory of Financial Information Processing, Putian University, Putian 351100, China. E-mail: ZY_LoYe@126.com.*
- *Chunxiang Xu is with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, Sichuan 610056, China, and also with the Yangtze Delta Region Institute (Huzhou), University of Electronic Science and Technology of China, Huzhou 610056, China. E-mail: chxxu@uestc.edu.cn.*
- *Nan Cheng is with the Department of Telecommunication, Xidian University, Xi'an 710071, China. E-mail: dr.nan.cheng@ieee.org.*
- *Zhi Liu is with the Department of Computer and Network Engineering, University of Electro-Communications, Chofugaoka 182-8585, Japan. E-mail: liu@ieee.org.*
- *Xuemin Shen is with the Department of Electronic and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada. E-mail: sshen@uwaterloo.ca.*

Manuscript received 21 February 2022; revised 22 July 2022; accepted 5 August 2022. Date of publication 16 August 2022; date of current version 3 October 2023.
This work was supported in part by the National Key R&D Program of China under Grant 2021YFB3101100, in part by the National Nature Science Foundation of China under Grant 62002050, in part by the Fujian Key Laboratory of Financial Information Processing (Putian University) under Grant JXC202202, in part by Key R&D Program of Sichuan under Grant 2021YFG0158, in part by Sichuan Science and Technology Program under Grant 2020JDTD0007, and in part by Central University Basic Research Funds Foundation under Grants A030202063008083 and ZYGX2020ZB027. (Corresponding author: Yuan Zhang.)
Digital Object Identifier no. 10.1109/TMC.2022.3199048

called "conditional forward transparency" and would be detailed in Section 3.2. In addition, a patient always needs to (delegate a doctor to) access parts of her/his EHRs. From the perspective of practicality as well as security protection, there should be no excessive costs and the key management problem introduced on the patient side to access any part of EHRs. Such a requirement is called "portability", which indicates that (a) all operations on the patient side can be performed by using a resource-constrained device (e.g., a smartphone) and (b) the patient can execute the prescribed scheme using any device she/he can access (i.e., the patient does not need to maintain any secret on her/his local device).

Furthermore, directly integrating existing data provenance mechanisms [21], [22], [23], [24], [25], [26] into cloud-based eHealth systems could not be satisfactory due to the lack of appropriate data structure of the provenance record. Specifically, as regards privacy, in general provenance mechanisms, privacy protection of personal information is not considered in designing the data structure of provenance records. Consequently, adopting these mechanisms in cloud-based eHealth systems would cause the privacy leakage from provenance records. As regards functionality, in general provenance schemes, the data structure of provenance records cannot accurately reflect the provenance information about EHRs. Therefore, in addition to the requirements on security and functionality of traditional cloud storage systems, more specific requirements on healthcare applications should be analyzed and considered in developing secure cloud-based eHealth systems.

In this paper, we provide a comprehensive analysis of the requirements of healthcare applications from the aspects of security, efficiency, and functionality. According to the analysis, we enumerate the primary requirements of cloud-based eHealth systems. We propose two security mechanisms to satisfy these requirements: a servers-aided password-based subsequent-key-locked encryption mechanism and a blockchain-based secure EHR provenance mechanism. Based on the two mechanisms, we develop a cloud-based eHealth system, dubbed HealthFort, with conditional forward transparency, portability, and secure EHR provenance. The following is the summary of the main contributions of this paper:

1) We investigate actual cloud-based eHealth systems and enumerate their primary requirements: EHR confidentiality, conditional identity privacy preservation, conditional forward transparency, backward security, provenance record security, portability, and efficient data access and auditing. We will detail them in Section 3.2.
2) We propose a servers-aided password-based subsequent-key-locked encryption mechanism that integrates the symmetric-key encryption [27] and a servers-aided password-hardening protocol [7], [28], [29]. Therefore, the proposed encryption mechanism is highly-efficient and robust against off-line dictionary guessing attacks (DGA). It enables the patients to authenticate themselves with the cloud server using their passwords as the *sole* factor and to permit the doctor to view their specific parts of outsourced EHRs using only one encryption key (at most 256 bits).
3) We propose a blockchain-based secure provenance mechanism for EHRs, where (a) the data structure of provenance records is designed to precisely reflect the provenance information about the underlying EHRs during their lifecycle and (b) a smart contract is deployed on a public blockchain (e.g., Ethereum) to assure the security of provenance records. With the provenance mechanism, both the outsourced EHRs and the corresponding provenance records are securely maintained and can be efficiently audited.
4) We integrate the proposed two mechanisms into one system, dubbed HealthFort, and develop it to provide a secure EHR storage service with efficient provenance. We provide formal security analyses to prove that HealthFort meets the security requirements. We implement a prototype of HealthFort and present the implementation details. Based on the prototype, we give a comprehensive evaluation to show the high efficiency of HealthFort.

The remainder of this paper is organized as follows. In Section 2, we review the related work and summarize the changes between HealthFort and the primary work, BESURE [1]. In Section 3, we present the problem statement. We overview HealthFort and present its construction in Sections 4 and 5, respectively. In Section 6, we analyze the security of HealthFort in terms of EHRs confidentiality and provenance security. In Section 7, we implement HealthFort and evaluate its performance. In Section 8, we draw the conclusion and outlook the future research directions.

2 Related Work

2.1 EHRs Security in Cloud-Based eHealth Systems

Cloud-based eHealth systems provide a convenient and efficient way to manage EHRs. It not only frees medical institutions from deploying and maintaining local storage devices [30], but also enables EHRs to be accessible in different devices and locations. However, such an outsourcing paradigm also introduces new threats towards the security of EHRs [31], [32], [33].

In recent years, EHRs security has become a major concern to both governments and individuals. For governments, several laws and regulations have been proposed. For instance, the Health Insurance Portability and Accountability Act (HIPAA) [34] indicates that EHR confidentiality is a fundamental requirement rather than an optional one; General Data Protection Regulation (GDPR) indicates that personal sensitive data, such as health data is subject to a higher level of protection. For individuals, more and more patients turn their attention to the security of their EHRs[1]. Typically, there are 13 patient data breaches in the past year that have resulted in lawsuits filed by patients[2]. As patients demand increased security for their EHRs, many protection mechanisms have been proposed [12], [16], [35], [36], [37], [38], [39].

1. General data protection regulation. https://gdpr- info.eu
2. https://www.beckershospitalreview.com/cybersecurity/13-patient-data-breach-lawsuits-in-the-past-year.html

However, in existing works, several practical requirements of actual eHealth systems are not considered. For example, in traditional encryption mechanisms for EHRs, a patient has to send multiple decryption keys to enable a doctor to access only a specific subset of the entire EHRs for diagnosing [12], [16], [35], [40], [41], which introduces heavy communication overhead on the patient side. Although the attribute-based encryption (ABE) technique [32], [42], [43] can be utilized to resolve this tension, it suffers from the single-point-of-failure problem and causes heavy computation overhead on both the patient and cloud server sides.

Moreover, in existing eHealth systems [36], [38], a patient needs to manage the decryption keys of EHRs locally. To free the patient from maintaining the decryption keys, a layered encryption mechanism could be utilized [44], where a patient maintains a master key locally, encrypts the decryption keys of EHRs under the master key, and outsources the ciphertexts of the decryption keys and the encrypted EHRs to a cloud server. By doing so, the patient does not need to manage the decryption keys of EHRs in her/his local devices. Nevertheless, in such a layered mechanism, the key management problem on the patient side still exists: the patient has to well maintain the master key for retrieving outsourced EHRs. A feasible way to address the key management problem is to use a password-based layered encryption technique: the patient encrypts the encryption keys under a human-memorable password instead of the master key [7], [29]. However, due to the low entropy of human-memorable passwords, this mechanism is vulnerable to DGA.

2.2 Secure Data Provenance in Cloud Storage

Data provenance has become an essential component for cloud storage systems, where the outsourced data is under threats from various aspects. Data provenance is a technique that describes the history of the actions performed on some data during the lifecycle [45], which supports data auditing and post-investigation.

In [46], Lynch first pointed out the necessity and importance of data provenance in digital systems. Muniswamy-Reddy et al. then observed that the data provenance technique is crucial for cloud storage systems [18], [19]. These works mainly focus on how to enhance the security of cloud storage systems using the data provenance technique. However, the security of the data provenance technique itself is not considered. If the provenance information can be arbitrarily modified, it is not useful in reality.

Hasan et al. [47] first identified the main challenges in trustworthy of provenance information and defined the main factors that impact the security of provenance. The first formal definition and security notions of data provenance for cloud storage systems were proposed by Lu et al. [22], where the basic requirements of a secure provenance scheme for outsourced data are first analyzed. Subsequently, several schemes with improvements on security and efficiency were proposed [23], [48], [49]. However, all these schemes rely on a trusted identity manager to ensure the security of the provenance information and are confronted with the single-point-of-failure problem.

To address the single-point-of-failure problem, subsequent works [21], [24], [26] utilized blockchain as the key technique: the provenance information is recorded to the blockchain such that adversaries cannot tamper with it, even if the adversary colludes with the cloud server. However, these schemes cannot be directly utilized in cloud-based eHealth systems, since the underlying data structure of provenance records cannot precisely reflect the EHRs' provenance and some specific requirements of actual healthcare applications are not considered.

2.3 Compared With BESURE

In the previous version of this paper [1], BESURE, a blockchain-based cloud-assisted eHealth system with secure data provenance, has been proposed. In BESURE, a servers-aided password-based subsequent-key-locked encryption framework is used to ensure the confidentiality of EHRs and the conditional forward transparency, a blockchain-based secure data provenance mechanism is used to track all operations performed on EHRs during their lifecycle. Compared with BESURE [1], *we have made the following improvements in this paper*.

- Concrete instantiation. We instantiate the servers-aided password-based subsequent-key-locked encryption mechanism based on CBC-mode of block cipher. The ciphertext of a key is generated by inputting the XOR of the plaintext of the current key and the ciphertext of the previous key into a pseudorandom function.
- Security analysis. We provide a formal security analysis for the proposed encryption mechanism, which proves that it is indistinguishable under the chosen-plaintext attack (IND-CPA).
- Prototype implementation. We implement a HealthFort prototype and present the implementation details. We evaluate the performance of HealthFort in a comprehensive way and compare HealthFort with existing systems in terms of security, functionality, and efficiency, which shows the high efficiency and advantage of HealthFort.

3 Problem Statement

3.1 Notation and Basic Theory

In this paper, ℓ denotes the security parameter; given two bit strings st_1 and st_2, $st_1||st_2$ denotes their concatenation; $z \leftarrow F(\cdot)$ denotes that z is an output of the function $F(\cdot)$; $i++$ denotes that $i = i + 1$; $r \xleftarrow{\$} S$ denotes randomly choosing r from the set S.

Bilinear pairing. Let G be an additive group with a primer order p, G_T is a multiplicative group with the same order as G. e is a bilinear pairing if it has the following three properties: *Bilinearity*: $\forall\ X, Y \in G, \forall\ v, w \in Z,\quad e(vX, wY) = e(X, Y)^{vw}$; *Non-degeneracy*: $\forall\ X, Y \in G, X \neq Y, e(X, Y) \neq 1$; *Efficiency*: $\forall\ X, Y \in G$, exists an efficient algorithm to compute $e(X, Y)$.

Threshold Cryptosystem. In a (t, n)-threshold cryptosystem, a secret s is split into n secret shares and each secret share is kept by a participant. Any t participants can pool their shares and perform certain cryptographic operations (e.g., generating signatures, encryption/decryption), but a

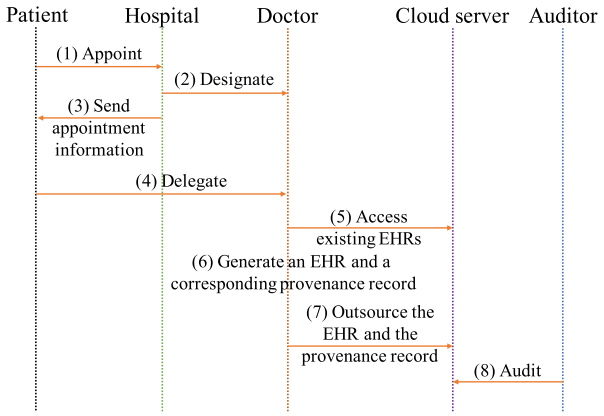


Fig. 1. Data flow of eHealth systems with EHR provenance.

collusion of less than t participants cannot extract any information about s from their shares.

Pseudorandom Function. Let F: $\{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}^*$ be an efficient, length-preserving, keyed function. F is a pseudorandom function if for all probabilistic polynomial-time (PPT) $\mathcal{M}$, there is a negligible function negl s.t.

$$|\Pr[\mathcal{M}^{\mathrm{F}_k(\cdot)(1^\ell)} = 1] - \Pr[\mathcal{M}^{f(\cdot)}(1^\ell) = 1]| \leq \mathrm{negl}(\ell),$$

where $k \xleftarrow{\$} \{01\}^\ell$, $f \xleftarrow{\$} \mathrm{Func}_\ell$, and Func_ℓ is the set of all functions mapping ℓ-bit strings to ℓ-bit strings.

Blockchain and Smart Contract. A blockchain is essentially a distributed ledger maintained by a group of network nodes and records the history of "transactions". Each node can share and verify the transaction data on the blockchain, but no one can fully control the data to be recorded to the blockchain [50]. The security of the blockchain ensures that any "efficient" adversary cannot tamper with the data recorded on the blockchain. Here, the definition of "efficient adversary" depends on the underlying consensus algorithm.

In this paper, we construct HealthFort on the Ethereum blockchain [51], since it is more expressive than others. Due to space limitations, we would not elaborate on the construction of the Ethereum blockchain, please refer to Ref. [51], [52] for more details.

A smart contract is a computerized transaction protocol that executes the terms of a contract [53]. In Ethereum, once a contract is deployed, an account address is assigned to store the smart contract code and the data generated during the deployment and execution of the smart contract. A smart contract can be triggered by sending transactions to the address of its account. After being activated, miners would run the smart contract code according to the data in the "data field" of the corresponding transaction.

3.2 Analysis of Actual eHealth Systems

From the perspective of EHRs' processing, the procedure of a general cloud-based eHealth system with EHR provenance is depicted in Fig. 1 and described as follows. When a patient (say $\mathcal{P}$) needs to consult a doctor, $\mathcal{P}$ first makes an appointment to the hospital. The latter then delegates a specific doctor (say $\mathcal{D}$) and sends the appointment information (including the doctor information and treatment information) to $\mathcal{P}$. At the corresponding point in time, $\mathcal{P}$ consults $\mathcal{D}$ and generates a delegation information to $\mathcal{D}$. $\mathcal{D}$ may need to view the existing EHRs of $\mathcal{P}$ for diagnosis, which requires $\mathcal{D}$ (who has $\mathcal{P}$'s delegation) to interact with the cloud server (say $\mathcal{CS}$) to access the EHRs. After the consultation, $\mathcal{D}$ generates some new EHRs for $\mathcal{P}$ and outsources them to the cloud server. In addition, $\mathcal{D}$ needs to generate a corresponding provenance record to describe how the $\mathcal{P}$'s EHRs are updated and the activities that $\mathcal{D}$ has performed. $\mathcal{D}$ also outsources the provenance record to $\mathcal{CS}$. At any later point in time when a medical disputation occurs, an auditor (say $\mathcal{A}$) who is subject to an authority can launch data investigation on the outsourced EHRs as well as the provenance records.

The above procedure does not consider the security and privacy issues in reality. We analyze actual cloud-based eHealth systems and point out the requirements in terms of security and privacy below.

EHR Confidentiality. As EHRs include privacy information about their owners, it is very dangerous to store the EHRs in the plaintext form. To preserve patient's privacy, EHRs should be encrypted before storing. We stress that ensuring EHRs' confidentiality is not just required by patients themselves, it is also stipulated in laws and regulations of most of governments, e.g., HIPAA [34] and GDPR.

Conditional Identity Privacy Preservation. For a specific diagnosis, the doctor's identity is also sensitive information. If it can be extracted by the adversary, the patient's privacy could be easily violated. On the other hand, when a medical disputation occurs, the doctor should not be able to deny that he generated some EHRs for the patient. Therefore, for a general diagnosis, the doctor's identity should be well protected in the corresponding provenance record, but for a disputed diagnosis, the auditor can easily extract the doctor's identity from the provenance record in a non-repudiation way.

Conditional Forward Transparency. In actual eHealth systems, a doctor always needs to access the patient's previous EHRs for diagnosis, which requires that the patient's EHRs should be "forward-transparent" to the delegated doctor. In other words, for the delegated doctor, she/he can access the patient's outsourced EHRs as needed. However, the forward transparency should not cause the abuse of patient's EHRs. For example, in most cases, when a patient consults a doctor from the department of Orthopedics, the doctor should only able to access the patient's EHRs generated by the doctors from the same department, rather than those generated by the doctors from other departments (e.g., department of Gastroenterology). Therefore, the forward transparency to doctors should be conditional, any "over-access" should be further permitted by the patient.

Backward Security. For privacy protection, after the diagnosis, the doctor should not be able to further access the outsourced EHRs of the patient, i.e., the contents of subsequent EHRs generated by other doctors (even if they are from the same department) should also be protected against the doctor.

Provenance Record Security. Once a provenance record is generated, it should not be forged or deleted, even though the corresponding operation on EHRs is mistakenly performed. This guarantees that the provenance records are able to reflect what happens to the patient's EHRs in a clear and precise way.

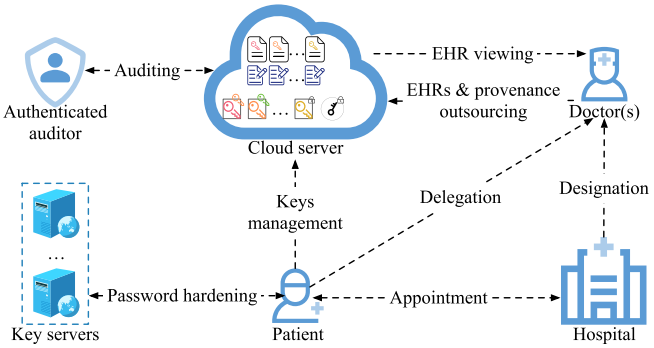


Fig. 2. System model.

From the perspective of practicality, there are some considerations as follows.

Portability. It is impractical to require patients to use specifically-crafted (powerful) devices in cloud-based eHealth systems, even if for online diagnoses. Hence, patients always desire an eHealth system to be friendly without sacrificing the security: they can access their EHRs anywhere utilizing any device while a strong security assurance still retains. We stress that the deployment of cloud storage systems does not address this problem, since patients need to keep a (long) secret key locally for authenticating themselves with the cloud server and decrypting the outsourced EHRs. Therefore, portability, which means that patients free from equipping specific (powerful) devices and do not need to maintain any secret locally, is an important requirement in actual eHealth systems.

Efficient Data Access and Auditing. In reality, the number of a patient's EHRs could be numerous. If the doctor wants to access a specific part of these EHRs, she/he should not need to download all of them. Moreover, as all provenance records need to be validated by the auditor, when the auditor checks the provenance records and EHRs for digital investigations, the checking should be completed within a short delay with low costs.

3.3 System Model

Based on the above analysis of cloud-based eHealth systems with EHR provenance, we construct HealthFort. As shown in Fig. 2, there are six different entities in HealthFort: doctors, patient, cloud server, hospital, key servers, and authenticated auditor.

Considering the portability on the patient side, in HealthFort, a patient only needs to use a password as the only secret for the whole consultation process. Nevertheless, due to the low entropy of the password, an adversary can easily recover the patient's password by DGA. Hence, in actual systems, straightforward use of passwords cannot ensure the security. HealthFort employs a servers-aided mechanism [7], [29] to harden patients' passwords against DGA.

3.4 Threat Model

In the threat model, we consider the auditor as a trusted entity, since the checking result generated by the auditor solely depends on the underlying EHRs and their provenance records. If the checking result is biased, anyone in the system could detect it.

Other entities (i.e., the cloud server, key servers, hospital, and doctor) could misbehave in HealthFort. For the hospital and doctor, *they are fully trusted only during the diagnosing period* and might misbehave after a diagnosis. EHRs and the corresponding provenance records are confronted with the following attacks.

Leakage of EHRs Contents. The cloud server may extract information about the contents of EHRs from the ciphertext of the EHRs and corresponding keys. The doctors may attempt to view EHRs that are generated subsequently or not related to the current consultation.

DGA From key Servers and Cloud Server. Due to the low entropy of patients' passwords, both the key servers and cloud server attempt to compromise patients' passwords. If they succeed, not only HealthFort, but also all other related systems would suffer from critical threats, since patients (i.e., users) always utilize sister passwords [54].

Forgery and Deletion Attacks on Provenance Records. The doctor may collude with the cloud server to forge or delete outsourced provenance records for profits.

Repudiation Attacks on Provenance Records. A doctor may deny that some provenance records are generated by her/him.

We suppose that a secure channel between a doctor and the hospital can be established by default, and there is a bound (i.e., a threshold determined by the security parameter) on the number of key servers that an adversary can compromise.

3.5 Design Goals

HealthFort should achieve following goals.

1) *Functionality*. HealthFort should provide a storage service for cloud-based eHealth systems with portability and EHR provenance.
2) *Security*. HealthFort should guarantee EHR confidentiality, conditional identity privacy preservation, conditional forward transparency, backward security, and provenance record security under the proposed threat model.
3) *Efficiency*. HealthFort should achieve high efficiency in terms of communication and computation on the patient, doctor, and auditor sides.

4 Overview of HealthFort

We overview HealthFort, focusing on the challenges addressed by this paper.

Ensuring EHR Confidentiality. EHRs' confidentiality is a fundamental requirement in cloud-based eHealth systems, where the encrypt-then-outsource paradigm has been widely deployed in reality. Traditional encryption (including symmetric-key encryption and public-key encryption) algorithms [12], [14], [32] can only ensure the confidentiality, but neither the portability on the patient side nor efficient conditional forward transparency can be achieved. A patient has to maintain at least one (long) secret key locally to (1) authenticate herself/himself with the cloud server and the hospital/doctor and (2) decrypt the outsourced EHRs as needed.

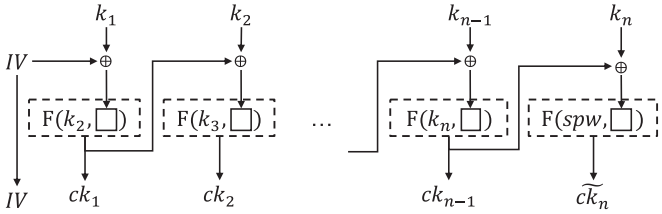


Fig. 3. Server-aided password-based subsequent-key-locked encryption.

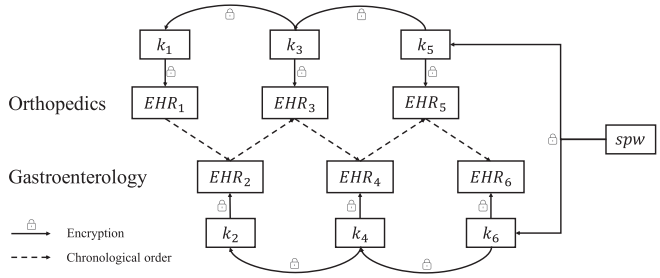


Fig. 4. The form of EHRs corresponding to multiple departments.

Portability. The key observation behind HealthFort is that passwords can serve as a fundamental role to achieve the portability on the patient side. Specifically, patients can authenticate themselves with the cloud server by only using their password; they can also encrypt all their EHRs with the password.

However, such a password-based protection mechanism has an inherent vulnerability that an adversary can easily compromise it by launching offline DGA. HealthFort addresses this problem by utilizing a servers-aided paradigm [28], [29]: the password is hardened with the aid of a set of key servers (which hold a server-side secret in a (t, n)-threshold way) before being used in reality, which protects the password from DGA while remaining its functionality. The interactions between the patient and key servers are oblivious such that for a specific password, the patient can compute a deterministic servers-hardened password but the key servers cannot extract anything about the password from the interactive message. Such a servers-hardened password is used for authentication between the patient and cloud server and for encrypting/decrypting the EHRs. By doing so, the only secret the patient needs to maintain is a human-memorable password, which achieves the portability on the patient side.

Conditional Forward Transparency and Backward Security. The above mechanism still fails to achieve efficient conditional forward transparency. If the doctor needs to access a part of the patient's EHRs, the patient has to download the entire EHR set from the cloud server, decrypt the target ones and sends the plaintexts to the doctor. Note that directly sending the password to the doctor would cause fully forward transparency and make the backward security impossible, which contradicts the design goals.

To address this problem, inspired by [27], we propose a servers-aided password-based subsequent-key-locked encryption, which is shown in Fig. 3, where $\mathrm{F}(k,)$ denotes a pseudorandom random function with a key k[3]. For the sake of brevity, we only discuss the single-department case, i.e., all these EHRs are generated by the doctors from the same department. To ensure the backward security, different EHRs should be encrypted under different keys randomly chosen by different doctors, these keys are denoted by $k_1, k_2, \cdots, k_n$, where the index of each key indicates the *chronological order*. To achieve the forward transparency, an encryption key is protected under its (in chronological order) subsequent key corresponding to the subsequent EHRs from the same department; the final key (i.e., the tail key) is protected under the servers-hardened password spw.

At any point in time, when a doctor (say the $i+1$th doctor $\mathcal{D}_{i+1}$ for $i \in [0, n-1]$) wants to access the patient's EHRs, the patient only needs to download the ciphertext of k_i from the cloud server, retrieve k_i by using her/his password and sends k_i to $\mathcal{D}_{i+1}$. The latter is able to obtain all previous keys $k_1, k_2, \cdots, k_{i-1}$ and further decrypt the ciphertexts to get all the previous EHRs.

We further stress that the forward transparency should be conditional, i.e., without the patient's permission, the doctor cannot access the EHRs that are generated by doctors from other departments. Therefore, in reality, the patient's EHRs have the form as shown in Fig. 4.

Secure EHR Provenance. To secure EHR provenance, two challenges should be addressed. The first one is to propose a data structure of EHRs' provenance records. The second one is to design a security mechanism to protect provenance records from various attacks.

Algorithm 1. SC

```
1: i = 0;
2: data[ ][ ];
3: function Store(ID_P, per_D, _data):
   // per_D denotes a permit generated by P for D
4:   if (per_D is valid):
5:     data[ID_P][i]=_data;
6:     i + +;
7:   end if;
8: function Audit(ID_P, i):
9:   return data[ID_P][i];
```

To address the first challenge, we investigate EHRs from actual eHealth systems and extract provenance requirements from them. Based on W3C PROV Ontology [55], we propose a data structure of provenance records for EHRs.

To address the second challenge, we propose a secure EHR provenance mechanism based on a smart contract that is deployed on a public blockchain (i.e., Ethereum). The key technique behind the mechanism is to store all EHR provenance records in chronological order by using an authenticated way in the contract storage, which enables the auditor to check the validity of all EHR provenance records with minimal costs (only one signature verification and several hash computations) without the single-point-of-failure problem.

We integrate all above mechanisms and implement a prototype called HealthFort, which provides secure EHRs storage and provenance for cloud-based eHealth systems.

3. We assume that AES is a pseudorandom function and replace F with AES in our construction.

TABLE 1
Summary of Notations

Notation	Explanation
e	The bilinear pairing $e : G \times G \rightarrow G_T$
P	A generator of G
p	The order of G and G_T
H_1, H_2	Hash functions mapping into G
h	A hash function mapping into $0, 1^\ell$
$\hbar$	A hash function mapping into Z_p
$\mathrm{E}(K, M)$	A symmetric-key encryption algorithm takes as input a key K and a plaintext M
$\mathrm{Enc}(epk, M)$	A public-key encryption algorithm takes as input a key epk and a plaintext M
$\mathrm{Sig}(ssk, M)$	A signature algorithm takes as input a key ssk and a message M
n	The number of key servers
t	A threshold determined by ℓ
$sk_{\mathcal{P}}$	$\mathcal{P}$'s private key randomly chosen from Z_p^*
$sk_{\mathcal{H}}$	$\mathcal{H}$'s private key randomly chosen from Z_p^*
$pk_{\mathcal{P}}$	$\mathcal{P}$'s public key computed as $sk_{\mathcal{P}} \cdot P$
$pk_{\mathcal{H}}$	$\mathcal{H}$'s public key computed as $sk_{\mathcal{H}} \cdot P$
SC	A smart contract shown in Algorithm 1
Add	The address of SC on Ethereum

5 Proposed HealthFort

A patient $\mathcal{P}$ with identity $ID_{\mathcal{P}}$, a doctor $\mathcal{D}$ with identity $ID_{\mathcal{D}}$, a hospital $\mathcal{H}$ with identity $ID_{\mathcal{H}}$, a cloud server $\mathcal{CS}$ with identity $ID_{\mathcal{CS}}$, a set of key servers $\{\mathcal{KS}_1, ..., \mathcal{KS}_n\}$ with identities $\{ID_{\mathcal{KS}_1}, ..., ID_{\mathcal{KS}_n}\}$ and an authenticated auditor $\mathcal{A}$ are involved in HealthFort. Practically, there are many doctors in the system. For the sake of brevity, we only show the single-doctor case. Other doctors would follow the same process. We provide the notations used in this section and the corresponding explanations in Table 1.

Phase 1. EHR Generation.

Setup. Based on the security parameter ℓ, system parameters $\{p, P, G, G_T, e, H_1, H_2, h, \hbar, \mathrm{E}(\cdot), \mathrm{Enc}(\cdot), \mathrm{Sig}(\cdot), n, t\}$ are determined, SC is deployed on Ethereum.

Registration. In this algorithm, $\mathcal{P}$ registers with $\mathcal{KS}_i$ $(i \in [1, n])$, $\mathcal{CS}$, and $\mathcal{H}$. This algorithm is depicted in Fig. 5 and detailed as follows.

- $\mathcal{P}$ creates a password $pw_{\mathcal{P}}$ and sends $ID_{\mathcal{P}}$ to $\mathcal{KS}_i, i = 1, 2, ..., n$.
- For $i = 1, 2, ..., n$, if $ID_{\mathcal{P}}$ exists in $\mathcal{KS}_i$'s local storage, it means that $ID_{\mathcal{P}}$ has been registered; Otherwise, all key servers jointly execute Algorithm 2 to generate and share a server-side secret s in a distributed way.
- $\mathcal{P}$ requests a servers-hardened password $spw_{\mathcal{P}}$ by executing Algorithm 3.

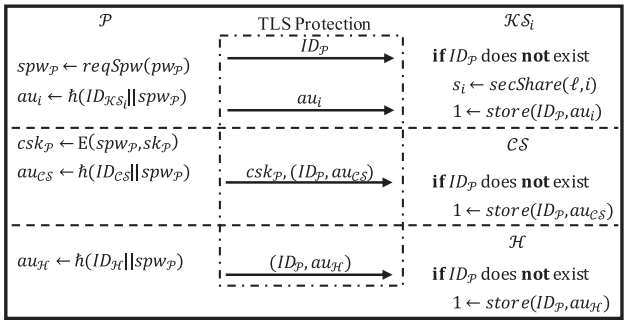


Fig. 5. Description of registration.

- $\mathcal{P}$ computes au_i and sends au_i to $\mathcal{KS}_i, i = 1, ..., n$ through a secure channel. $\mathcal{P}$ encrypts $sk_{\mathcal{P}}$ under $spw_{\mathcal{P}}$ to obtain $csk_{\mathcal{P}}$ and sends $\{csk_{\mathcal{P}}, (ID_{\mathcal{P}}, au_{\mathcal{CS}})\}$ to $\mathcal{CS}$. $\mathcal{P}$ computes $au_{\mathcal{H}}$ and sends $(ID_{\mathcal{P}}, au_{\mathcal{H}})$ to $\mathcal{H}$.
- If $\mathcal{P}$ did not register with $\mathcal{KS}_i$, $\mathcal{CS}$, and $\mathcal{H}$ before, $\mathcal{KS}_i$, $\mathcal{CS}$, and $\mathcal{H}$ store $(ID_{\mathcal{P}}, au_i)$, $(ID_{\mathcal{P}}, au_{\mathcal{CS}})$, and $(ID_{\mathcal{P}}, au_{\mathcal{H}})$, respectively, as the authentication credential. $\mathcal{CS}$ stores $csk_{\mathcal{P}}$.

Algorithm 2. *secShare*

1: Each $\mathcal{KS}_i$ $(i \in [1, n])$ randomly chooses $a_{i,0} \in Z_p^*$ and a polynomial $f_i(x)$ over Z_p with degree at most $t-1$, such that $f_i(0) = a_{i,0}$, where $f_i(x) = a_{i,0} + a_{i,1}x + ... + a_{i,t-1}x^{t-1}$.
2: Each $\mathcal{KS}_i$ $(i \in [1, n])$ calculates $a_{i,\gamma}P$, and $f_i(j)$, then sends them to $\mathcal{KS}_j$ $(\gamma = 0, 1, ..., t-1, j = 1, 2, ..., n, j \neq i)$.
3: After receiving $f_j(i)$, and $a_{j,\gamma}P$ $(j \in [1, n], j \neq i; \gamma \in [0, t-1])$, $\mathcal{KS}_i$ verifies $f_j(i)$, if Eq. (1) holds, $\mathcal{KS}_i$ accepts $f_j(i)$, otherwise, $\mathcal{KS}_i$ rejects.

$$f_j(i)P = \sum_{\gamma=0}^{t-1} i^\gamma \cdot a_{j,\gamma}P \tag{1}$$

4: $\mathcal{KS}_i$ calculates the server-side key secret share $s_i = \sum_{j=1}^{n} f_j(i)$ and the corresponding public share $Q_i = s_i P$.
5: The server-side secret key $s = \sum_{i=1}^{n} a_{i,0}$ is shared among $\mathcal{KS}_1, ..., \mathcal{KS}_n$ in a (t, n)-threshold secret-sharing way. The public servers-hardened key is $Q = sP$.

Appointment. As depicted in Fig. 6, $\mathcal{P}$ makes an appointments with $\mathcal{H}$, and $\mathcal{H}$ designates $\mathcal{D}$ for $\mathcal{P}$.

- $\mathcal{P}$ executes Algorithm 3 to request $spw_{\mathcal{P}}$ from $\mathcal{KS}_i$.
- $\mathcal{P}$ computes $au_{\mathcal{H}}$. With $(ID_{\mathcal{P}}, au_{\mathcal{H}})$, $\mathcal{P}$ and $\mathcal{H}$ can establish a secure channel.[4] Using this channel, $\mathcal{P}$ can authenticate with $\mathcal{H}$, and the following interactions are in this channel.

4. After a successful authentication, the transmission between $\mathcal{P}$ and $\mathcal{H}$ can be protected by authenticated encryption (e.g., AES-GCM), where the symmetric keys are derived by $au_{\mathcal{H}}$.

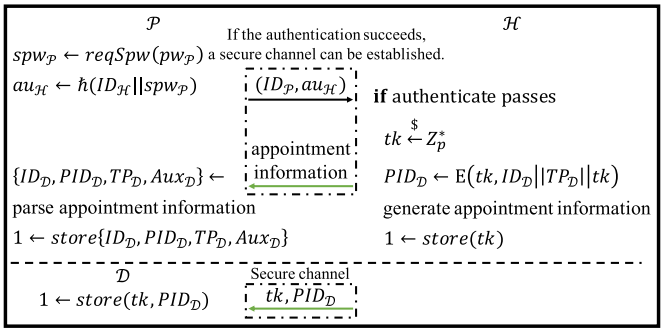


Fig. 6. Description of appointment.

- $\mathcal{H}$ chooses a treatment key $tk \in Z_p^*$ randomly, generates a pseudonym $PID_{\mathcal{D}}$ for $\mathcal{D}$ using tk, sends appointment information to $\mathcal{P}$, and sends $\{PID_{\mathcal{D}}, tk\}$ to $\mathcal{D}$ through a secure channel.
- $\mathcal{H}$ stores tk locally, which is used to attach $ID_{\mathcal{D}}$ to the pseudonym $PID_{\mathcal{D}}$ during the audit.
- $\mathcal{P}$ obtains $ID_{\mathcal{D}}$, the validity period $TP_{\mathcal{D}}$, and other auxiliary information $Aux_{\mathcal{D}}$, such as the department De of $\mathcal{D}$ and tk from the appointment information.
- $\mathcal{D}$ maintains $\{PID_{\mathcal{D}}, tk\}$ locally for the subsequent consultation and provenance generation.

Algorithm 3. *reqSpw*

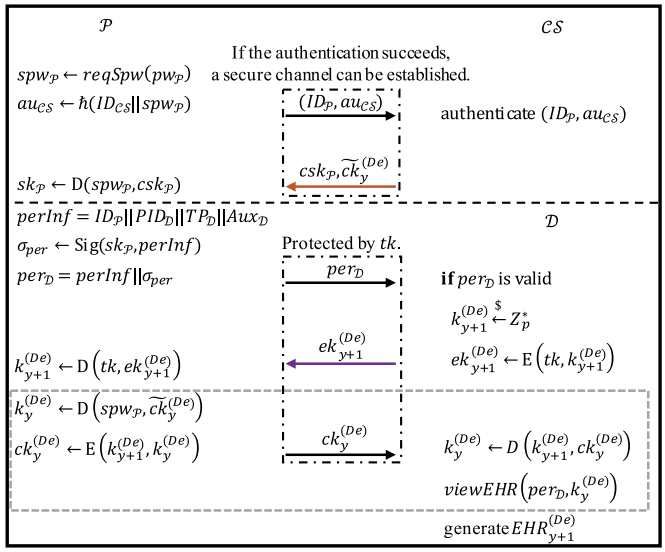
1: $\mathcal{P}$ chooses $r \in Z_p^*$ randomly, blinds $pw_{\mathcal{P}}$ by computing $pw_{\mathcal{P}}^* = r \cdot H_1(pw_{\mathcal{P}})$, and sends $(ID_{\mathcal{P}}, pw_{\mathcal{P}}^*)$ to $\mathcal{KS}_i$ $(i \in [1, n])$.
2: After receiving $(ID_{\mathcal{P}}, pw_{\mathcal{P}}^*)$, each $\mathcal{KS}_i (i \in [1, n])$ computes $\sigma_i^* = s_i \cdot pw_{\mathcal{P}}^*$, and sends σ_i^* to $\mathcal{P}$.
3: $\mathcal{P}$ verifies σ_i^* by $e(\sigma_i^*, P) \stackrel{?}{=} e(pw_{\mathcal{P}}^*, Q_i)$. If σ_i^* is valid, $\mathcal{P}$ accepts it.
4: After receiving t valid signatures, $\mathcal{P}$ computes σ_{pw} by

$$\sigma_{pw} = r^{-1} \cdot \sum_{l=1}^{t} \left(\prod_{\substack{1 \le \epsilon \le t \\ \epsilon \ne l}} \frac{\epsilon}{\epsilon - l} \right) \cdot \sigma_l^*,$$

and obtains $spw_{\mathcal{P}}$ by $spw_{\mathcal{P}} = \hbar(\sigma_{pw}||pw_{\mathcal{P}})$.

Consultation. In this algorithm, $\mathcal{P}$ consults $\mathcal{D}$, authorizes $\mathcal{D}$ to access parts of EHRs related to the current diagnosis. $\mathcal{D}$ creates a new EHR for $\mathcal{P}$. The interactions between $\mathcal{P}$ and $\mathcal{D}$ are protected by tk. A schematic diagram is shown in Fig. 7, where the steps in the gray dotted box are executed when $\mathcal{D}$ needs to access existing EHRs of $\mathcal{P}$.

- $\mathcal{P}$ takes $pw_{\mathcal{P}}$ as input to request $spw_{\mathcal{P}}$ from $\mathcal{KS}_i$ $(i = 1, 2, ..., n)$ by executing Algorithm 3 and computes $au_{\mathcal{CS}}$.
- With $(ID_{\mathcal{P}}, au_{\mathcal{CS}})$, a secure channel can be established. Using this channel, $\mathcal{P}$ can authenticate with $\mathcal{CS}$ and download $csk_{\mathcal{P}}$ from $\mathcal{CS}$, decrypts $csk_{\mathcal{P}}$ using $spw_{\mathcal{P}}$ to get $sk_{\mathcal{P}}$.
- $\mathcal{P}$ generates a permission $per_{\mathcal{D}}$ to delegate $\mathcal{D}$, $per_{\mathcal{D}}$ includes permission data and the corresponding signature, and sends $per_{\mathcal{D}}$ to $\mathcal{D}$.
- If $per_{\mathcal{D}}$ is valid, $\mathcal{D}$ accepts it and randomly chooses $k_{y+1}^{(De)} \in Z_p^*$ as the encryption key on the EHR generated in this diagnosis, encrypts it to $ek_{y+1}^{(De)}$ using tk, and sends the ciphertext to $\mathcal{P}$.



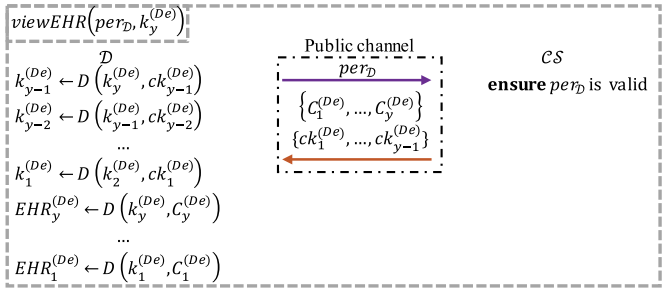


Fig. 7. Description of consultation.

- $\mathcal{P}$ decrypts $ek_{y+1}^{(De)}$ to obtain $k_{y+1}^{(De)}$.

If $\mathcal{D}$ needs to view $\mathcal{P}$'s EHRs that have been outsourced before for reference, $\mathcal{D}$ requests a permission to access these EHRs from $\mathcal{P}$ as follows.

- $\mathcal{P}$ downloads $\tilde{ck}_y^{(De)} = \mathrm{E}(spw_{\mathcal{P}}, k_y^{(De)})$ from $\mathcal{CS}$, where $k_y^{(De)}$ is the tail of the corresponding key chain. $\mathcal{P}$ decrypts the ciphertext to obtain $k_y^{(De)}$, computes $ck_y^{(De)}$, and sends it to $\mathcal{D}$.
- $\mathcal{D}$ obtains $k_y^{(De)}$ by decrypting $ck_y^{(De)}$ and sends $per_{\mathcal{D}}$ to $\mathcal{CS}$ for authentication.
- $\mathcal{CS}$ verifies the validity of $per_{\mathcal{D}}$ to check whether the doctor is delegated by the patient. If the verification passes, $\mathcal{CS}$ allows $\mathcal{D}$ to download the ciphertexts of the EHRs from De and the ciphertexts of the corresponding encryption keys.
- $\mathcal{D}$ can obtain $k_{y-1}^{(De)}, k_{y-2}^{(De)}, ..., k_1^{(De)}$ by decrypting $ck_{y-1}^{(De)}, ck_{y-2}^{(De)}, ..., ck_1^{(De)}$ successively using $k_y^{(De)}$, further decrypt the ciphertexts of EHRs. By doing so, $\mathcal{D}$ can view EHRs from De.
- Finally, $\mathcal{D}$ creates $EHR_{y+1}^{(De)}$ for $\mathcal{P}$ and encrypts $EHR_{y+1}^{(De)}$ as $C_{y+1}^{(De)} = \mathrm{E}(k_{y+1}^{(De)}, EHR_{y+1}^{(De)})$.

Phase 2. Provenance generation.

The procedure of *Phase 2* is shown in Fig. 8.

ProvGen. In this algorithm, $\mathcal{D}$ generates a provenance record PR_ι and requests a signature on PR_ι from $\mathcal{H}$. The structure of a provenance record is given in Fig. 9, where exd, exp, exh denote the specific namespace prefixes used by De, $\mathcal{P}$, and $\mathcal{H}$, respectively.

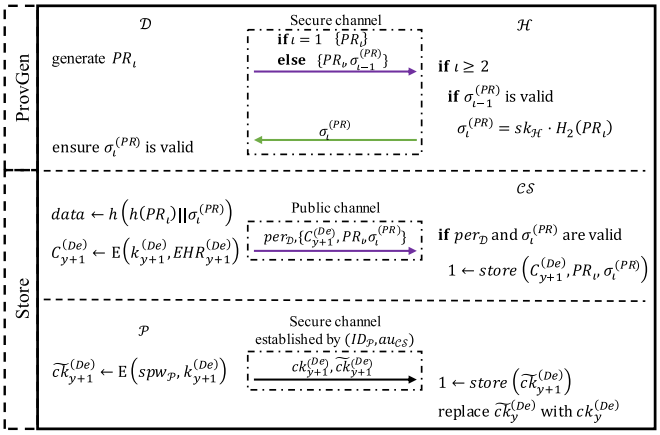


Fig. 8. Description of phase 2. Provenance generation.

- $\mathcal{D}$ generates a provenance record PR_ι as shown in Fig. 9.
- If $EHR_{y+1}^{(De)}$ is $\mathcal{P}$'s first EHR (i.e., $\iota = 1$), EHRViewActivity and EHR_{Vi} fields are set to "NULL"; if it is the first time that $\mathcal{P}$ comes to this department (i.e., $y + 1 = 1$), EHRViewActivity is set to "NULL".
- For $\iota = 1$, $\mathcal{D}$ sends PR_ι to $\mathcal{H}$ via a secure channel; for $\iota \geq 2$, $\mathcal{D}$ sends $\{PR_\iota, \sigma_{\iota-1}^{(PR)}\}$ to $\mathcal{H}$ through a secure channel, where $\sigma_{\iota-1}^{(PR)}$ is a signature generated by $\mathcal{H}$ on the previous provenance record.
- If $\iota \geq 2$, $\mathcal{H}$ verifies $\sigma_{\iota-1}^{(PR)}$. If the verification passes, $\mathcal{H}$ signs on PR_ι as $\sigma_\iota^{(PR)} = sk_{\mathcal{H}} \cdot H_2(PR_\iota)$ and sends $\sigma_\iota^{(PR)}$ to $\mathcal{D}$.
- If $\sigma_\iota^{(PR)}$ is valid, $\mathcal{D}$ accepts it as the signature of $\mathcal{H}$ on PR_ι.

Store. In this algorithm, $\mathcal{D}$ outsources $EHR_{y+1}^{(De)}$ and the corresponding provenance information to $\mathcal{CS}$.

- $\mathcal{D}$ computes $data = h(h(PR_\iota)||\sigma_\iota^{(PR)})$, invokes $SC.Store(ID_{\mathcal{P}}, per_{\mathcal{D}}, data)$ to integrate the provenance information into the blockchain by sending the transaction Tx_ι to *Add* as shown in Fig. 10.
- After Tx_ι is recorded into the blockchain, $\mathcal{D}$ sends $\{PR_\iota, \sigma_\iota^{(PR)}, C_{y+1}^{(De)}, per_{\mathcal{D}}\}$ to $\mathcal{CS}$.
- $\mathcal{CS}$ accepts and stores $\{PR_\iota, \sigma_\iota^{(PR)}, C_{y+1}^{(De)}, per_{\mathcal{D}}\}$ if $per_{\mathcal{D}}$ and $\sigma_\iota^{(PR)}$ are valid.
- $\mathcal{P}$ computes $\tilde{ck}_{y+1}^{(De)} = \mathrm{E}(spw_{\mathcal{P}}, k_{y+1}^{(De)})$ and outsources $\{\tilde{ck}_{y+1}^{(De)}, ck_y^{(De)}\}$ to $\mathcal{CS}$. ($k_{y+1}^{(De)}$ has been sent to $\mathcal{P}$ as shown in Fig. 7.)
- $\mathcal{CS}$ stores $\tilde{ck}_{y+1}^{(De)}$ and replaces $\tilde{ck}_y^{(De)}$ with $ck_y^{(De)}$.

Finally, $\mathcal{P}$'s EHRs corresponding to De and the provenance information have the following form:

$$< C_1^{(De)}, C_2^{(De)}, ..., C_{y+1}^{(De)} >,$$
$$< \{PR_1, \sigma_1^{(PR)}\}, \{PR_2, \sigma_2^{(PR)}\}, ..., \{PR_\iota, \sigma_\iota^{(PR)}\} > .$$

Phase 3. Provenance verification.

Audit. Given the provenance information

$$< \{PR_1, \sigma_1^{(PR)}\}, \{PR_2, \sigma_2^{(PR)}\}, ..., \{PR_\iota, \sigma_\iota^{(PR)}\} >,$$

Data structure of a provenance record

1: exd: EHRGenerateActivity
   a prov: Activity;
   prov: startedAtTime "Start time of the activity";
   prov: endedAtTime "End time of the activity".
2: exd: EHR_{Ge} // The generated EHR.
   a prov: Entity;
   prov: wasGeneratedBy exd: EHRGenerateActivity;
   prov: generatedAtTime "Time of creating EHR_{Ge}";
   prov: value $h(C_{Ge})$. // A hash value of the ciphertext of EHR_{Ge}.
3: exd: EHRViewActivity
   a prov: Activity;
   prov: startedAtTime "Start time of the activity";
   prov: endedAtTime "End time of the activity".
4: exd: EHR_{Vi} // The viewed EHR.
   a prov: Entity;
   prov: value $h(C_{Vi})$. // A hash value of the ciphertext of EHR_{Vi}.
5: exp: $\mathcal{P}$ //The patient.
   a foaf: Person, prov: Agent;
   foaf: openid "$ID_{\mathcal{P}}$".
6: exd: $\mathcal{D}$ // The doctor.
   a foaf: Person, prov: Agent;
   foaf: openid "$PID_{\mathcal{D}}$".
7: exd: De // The department, which can be encrypted under tk for privacy preservation.
   a foaf: Group, prov: Agent;
   foaf: name "Name of De";
   foaf: openid "ID_{De}".
8: exh: $\mathcal{H}$ // The hospital.
   a foaf: Organization, prov: Agent;
   foaf: name "Name of $\mathcal{H}$";
   foaf: openid "$ID_{\mathcal{H}}$".

Fig. 9. Data structure of a provenance record.

$\mathcal{A}$ can verify the validity as follows.

- $\mathcal{A}$ verifies PR_ι by checking

$$e(\sigma_\iota^{(PR)}, P) \stackrel{?}{=} e(H_2(PR_\iota), pk_{\mathcal{H}}).$$

- $\mathcal{A}$ extracts the data information from the blockchain by invoking $SC.Audit(ID_{\mathcal{P}}, \mu), \mu \in [1, \iota]$.

Transaction Tx_ι

From
$\mathcal{D}$*'s address*
To
SC's address
Value
0
Data
$SC.Store(ID_{\mathcal{P}}, per_{\mathcal{D}}, h(h(PR_\iota)\|\|\sigma_\iota^{(PR)}))$
Signature
A signature on this transaction under $\mathcal{D}$'s key

Fig. 10. A schematic diagram of Tx_ι.

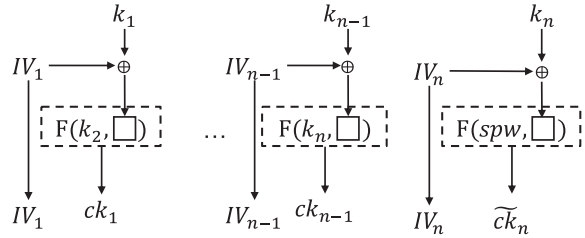


Fig. 11. The construction in the conference version.

- $\mathcal{A}$ computes $h(h(PR_\mu)||\sigma_\mu^{(PR)})$ for each $\mu = 1, 2, ..., \iota$, and verifies the integrity and timeliness of provenance by checking whether the provenance record's hash value matches the extracted data.
- If all the provenance records pass all the above checking, the provenance can be accepted.

6 Security Analysis

We analyze the security of HealthFort in terms of EHR confidentiality and provenance security.

6.1 EHR Confidentiality

In our servers-aided password-based subsequent-key-locked encryption mechanism, in the nth stage, a key k_i is encrypted under the subsequent key k_{i+1}, and the tail key k_n is encrypted under the hardened password spw. In each encryption, a random initialization vector is needed to provide the randomness, or it is vulnerable to CPA. Since an adversary can easily choose a challenge plaintext k_n that he has queried for the corresponding ciphertext under the hardened password spw.

Therefore, as shown in Fig. 11, the construction in BESURE (the conference version [1] of this paper) is choosing a fresh initialization vector for each encryption, which has been proven to achieve EHR confidentiality in [1]. However, such a construction requires patients to carefully choose a random initialization vector for each encryption and thereby the costs for generating randomness increase linearly with the number of keys in a chain (i.e., $O(n)$).

To free patients from multiple choosing random initialization vectors, inspired by CBC mode, we utilize the previous ciphertext as the initialization vector in the current encryption. These two constructions are detailed in Table 2. By doing so, patients only need to choose one initialization vector for a key chain. The costs for generating randomness are independent of the number of keys in a chain (i.e., $O(1)$) without breaking the IND-CPA property of the servers-aided password-based subsequent-key-locked encryption mechanism. We prove its security as follows.

Intuitively, EHR confidentiality captures the property that even if a malicious cloud server obtains an encryption key of an EHR, the cloud server cannot do better than performing online password guessing attacks to retrieve the subsequent keys to further obtain contents of corresponding EHRs. In HealthFort, EHR confidentiality is achieved by the servers-aided password-based subsequent-key-locked encryption mechanism, where EHRs are encrypted under the corresponding random-chosen keys, a key is encrypted under the subsequent key, and the tail key is encrypted under the servers-hardened password.

TABLE 2
Two Constructions of the Servers-Aided Password-Based Subsequent-Key-Locked Encryption Mechanism

Previous construction	$IV_i \xleftarrow{\$} \{0,1\}^\ell, \forall\, i \in [1, n-1]$ $ck_i \leftarrow \mathrm{F}(k_{i+1}, IV_i \oplus k_i), \forall\, i \in [1, n-1]$ $\tilde{ck}_n \leftarrow \mathrm{F}(spw, IV_n \oplus k_n)$
Current construction	$IV \xleftarrow{\$} \{0,1\}^\ell$ $ck_1 \leftarrow \mathrm{F}(k_2, IV \oplus k_1)$ $ck_i \leftarrow \mathrm{F}(k_{i+1}, ck_{i-1} \oplus k_i), \forall\, i \in [2, n-1]$ $\tilde{ck}_n \leftarrow \mathrm{F}(spw, ck_{n-1} \oplus k_n)$

The security of the proposed mechanism relies on a password-hardening protocol [28], [56], which requires (on the premise that no more than threshold number of key servers are compromised by an adversary) (1) *Pseudorandomness.* The adversary cannot distinguish spw from a random string with a non-negligible probability, (2) *Unpredictability.* When pw is randomly chosen from a password space PW, given the interaction information between the patient $\mathcal{P}$ and a set of key servers $\mathcal{KS}_i\,(i = 1, ..., n)$ (who assist to harden the password), the adversary cannot obtain more information about the spw from the interaction than from a random string, and (3) *Obliviousness.* The adversary cannot extract more information about the password pw from spw than from a random string. These properties rely on the Gap Threshold One More Diffie-Hellman (Gap-TOMDH) assumption provided in the following.

Definition 1 (Gap Threshold One-More Diffie-Hellman assumption). *Let $C_{t',t}(q_1, ..., q_n)$ be the largest value of v such that there exists binary vectors $\vec{u_1}, ..., \vec{u_v} \in \{0,1\}^n$ such that each $\vec{u_i}$ has $t - t'$ number of 1 in it and $(q_1, ..., q_n) \geq \sum_{i=1}^{v} \vec{u_i}$ (all operations on vectors are component-wise integer operations). A solver $\mathcal{S}$ is given a group G with a prime order p and a generator P, a set $R = \{P_1, ..., P_n\}$, where $P_i \xleftarrow{\$} G$ for $i \in [1, n]$, access to oracles TOMDH$(\cdot,\cdot)$ for a random polynomial $f(x) = \sum_{i=0}^{t-1} a_i \cdot x^i$ over Z_p and DDH$(\cdot,\cdot,\cdot,\cdot)$. $\mathcal{S}$ only can output $a_0 \cdot P_j$ for $q+1$ different elements P_i in R with a negligible probability, where $q \geq C_{t',t}(q_1, ..., q_n)$ for $t' < t$, and q_i is the number of $\mathcal{S}$ queries to TOMDH$(i, \cdot)$.*

- TOMDH(i, Q_i): input $i \in [1, n]$ and $Q_i \in G$, output $f(i) \cdot Q_i$.
- DDH(P_i, P_j, Q_i, Q_j): input $P_i, P_j, Q_i, Q_j \in G$, output 1 iff $\exists\, m \in Z_p$ such that $P_j = mP_i \wedge Q_j = mQ_i$.

Formally, with the Gap-TOMDH assumption, we have Lemma 1 described below.

Lemma 1. *The hardened password spw is pseudorandom, unpredictable, and oblivious under the Gap-TOMDH assumption in the random oracle model.*

Proof. The proof appears in Section 3.3 of [56] and Appendix A of [28]. Here we would not repeat it. □

With **Lemma 1**, we treat the password-hardening protocol as an "ideal functionality", model spw as the answer of a random oracle with the input pw to capture these properties.

A formal definition of EHR confidentiality is as follows.

$\underline{\mathrm{EConf}^{cpa}_{\mathcal{CS}^*}(1^\ell, n, PW)}$

$1:\ Q := \emptyset$

$2:\ pw \xleftarrow{\$} PW$

$3:\ spw \leftarrow H\mathcal{O}(pw)$

$4:\ k_2, k_3, ..., k_n \xleftarrow{\$} Z_p^*$

$5:\ \mathbb{O} := (\mathrm{H}\mathcal{O}, \mathrm{Enc}\mathcal{O}^{(k)}, \mathrm{Enc}\mathcal{O}^{(spw)})$

$6:\ (k_{0,1}.k_{1,1}) \leftarrow \mathcal{CS}^{*\mathbb{O}}(\ell)$

$7:\ b \xleftarrow{\$} \{0,1\}$

$8:\ \{ck_{b,1}, ck_{b,2}, ..., ck_{b,n-1}\} \leftarrow \mathrm{Enc}\mathcal{O}^{(k)}(k_{b,1})$

$9:\ \tilde{ck}_{b,n} \leftarrow \mathrm{Enc}\mathcal{O}^{(spw)}(k_{b,n}, spw)$

$10:\ b' \leftarrow \mathcal{CS}^{*\mathbb{O}}(ck_{b,1}, ck_{b,2}, ..., ck_{b,n-1}, \tilde{ck}_{b,n})$

$11:$ **return** $b = b'$

$\underline{\mathrm{H}\mathcal{O}(1^\ell, pw)}$

$1:$ **if** $Q[pw] =$null

$2:\quad Q[pw] \xleftarrow{\$} Z_p$

$3:$ **return** $Q[pw]$

$\underline{\mathrm{Enc}\mathcal{O}^{(k)}(\ell, k_1)}$

$1:\ IV \xleftarrow{\$} \{0,1\}^\ell$

$2:\ \tilde{ck}_1 \leftarrow \mathrm{F}(spw, IV \oplus k_1), ck_1 \leftarrow \mathrm{F}(k_2, IV \oplus k_1)$

$3:\ \cdots$

$4:\ \tilde{ck}_{n-1} \leftarrow \mathrm{F}(spw, ck_{n-2} \oplus k_{n-1})$
$\quad ck_{n-1} \leftarrow \mathrm{F}(k_n, ck_{n-2} \oplus k_{n-1})$

$5:\ \tilde{ck}_n \leftarrow \mathrm{F}(spw, ck_{n-1} \oplus k_n)$

$6:$ **return** $(IV, \tilde{ck}_1, ck_1, ..., \tilde{ck}_{n-1}, ck_{n-1}, \tilde{ck}_n)$

$\underline{\mathrm{Enc}\mathcal{O}^{(spw)}(1^\ell, m)}$

$1:\ IV' \xleftarrow{\$} \{0,1\}^\ell$

$2:\ cm \leftarrow \mathrm{F}(spw, IV' \oplus m)$

$3:$ **return** (IV', cm)

Fig. 12. The game for EHR confidentiality.

Definition 2 (EHR confidentiality). *HealthFort achieves IND-CPA secure for encrypted EHRs iff for any probabilistic polynomial-time (PPT) adversary $\mathcal{CS}^*$, any polynomial number of keys n, and any password space PW, the following equation holds, where $q(\ell)$ is an upper bound on the number of queries $\mathcal{CS}^*$ makes to the oracles, $p(\ell)$ is a polynomial,*

$$|\Pr[\mathrm{EConf}^{cpa}_{\mathcal{CS}^*}(1^\ell, n, PW) = 1]| \leq \frac{q(\ell)}{|PW|} + \frac{1}{2} + \frac{p(\ell)}{2^\ell} + \mathrm{negl}(\ell).$$

As shown in Fig. 12, the adversary $\mathcal{CS}^*$ is actually a malicious cloud server, $\mathcal{CS}^*$ can choose an arbitrary encryption key and query $\mathbb{O}$ on the key to obtain its ciphertext and the ciphertexts of subsequent keys. Finally, $\mathcal{CS}^*$ chooses two keys $k_{0,1}, k_{1,1}$ with the same length as the challenge plaintexts and obtains the ciphertext of $k_{b,1}$ and the ciphertexts of subsequent keys, where b is randomly chosen from the set $\{0,1\}$. The goal of $\mathcal{CS}^*$ is to correctly guess the value of b, i.e., winning the game. We further stress that in the game, we allow the malicious cloud server to collude with several (less than the threshold number of) key servers.

Theorem 1. *If $F(\cdot)$ is a pseudorandom function, $\hbar(\cdot)$ is modeled as a random oracle, then HealthFort achieves IND-CPA secure for encrypted EHRs.*

Proof. The proof of Theorem 1 consists of two steps.

In the first step of the proof, we first define G_0, which is a game that is exactly the same as EConf except that spw is replaced with a random string. In EConf, the hash function is actually modeled as a random oracle $\mathrm{H}\mathcal{O}$. Specifically, $\mathrm{H}\mathcal{O}$ maintains a list $Q[\,]$ that is initially empty. When $\mathrm{H}\mathcal{O}$ receives a query pw from $\mathcal{CS}^*$, it first checks whether $Q[pw]$ exists in the list, if so, the corresponding value $Q[pw]$ is returned. Otherwise, a uniform string is chosen and returned, and the oracle stores $Q[pw]$. In such a case, if $\mathcal{CS}^*$ does not query the random oracle on the correct password, EConf is identical with G_0. If $\mathcal{CS}^*$ queries the random oracle on the challenge password and obtains the corresponding spw, $\mathcal{CS}^*$ can easily succeed in the experiment.

Let Same denote the event that the challenge password is the same as the one that $\mathcal{CS}^*$ queries $\mathrm{H}\mathcal{O}$. Then, we have

$$\begin{aligned}
&\Pr[\mathrm{EConf}^{cpa}_{\mathcal{CS}^*}(1^\ell, n, PW) = 1] \\
&\quad = \Pr[\mathrm{EConf}^{cpa}_{\mathcal{CS}^*}(1^\ell, n, PW) = 1 \wedge \mathsf{Same}] \\
&\qquad + \Pr[\mathrm{EConf}^{cpa}_{\mathcal{CS}^*}(1^\ell, n, PW) = 1 \wedge \overline{\mathsf{Same}}] \\
&\quad = \Pr[\mathsf{Same}] + \Pr[\mathrm{G}_0^{\mathcal{CS}^*}(1^\ell, n, PW) = 1] \\
&\quad \leq \frac{q(\ell)}{|PW|} + \Pr[\mathrm{G}_0^{\mathcal{CS}^*}(1^\ell, n, PW) = 1].
\end{aligned}$$

In the second step of the proof, we demonstrate that

$$\Pr[\mathrm{G}_0^{\mathcal{CS}^*}(1^\ell, n, PW) = 1] \leq \frac{1}{2} + \frac{p(\ell)}{2^\ell} + \mathrm{negl}(\ell).$$

Let G_1 be an experiment that is identical with G_0, except that the pseudorandom function $\mathrm{F}(\cdot)$ is replaced with a random function $f(\cdot)$ with the same domain and range. We prove that

$$|\Pr[\mathrm{G}_0(1^\ell, n, PW) = 1] - \Pr[\mathrm{G}_1(1^\ell, n, PW) = 1]| \leq \mathrm{negl}(n).$$

We use $\mathcal{CS}^*$ to construct a distinguisher $\mathcal{M}$ for the pseudorandom function $\mathrm{F}(\cdot)$. $\mathcal{M}$ is given oracle access to some function $\mathcal{O}$, and its goal is to determine whether the first of the returned strings is generated by $\mathrm{F}(\cdot)$ or the truly random function $f(\cdot)$.

In detail:

$\mathcal{M}$ is given input $(1^\ell, n, PW)$ and access to an oracle $\mathcal{O}$.

1) Run $\mathcal{CS}^*$. Whenever $\mathcal{CS}^*$ queries the oracles $\mathbb{O}$ on k_1, $\mathcal{M}$ queries $\mathcal{O}$, obtains a response and returns the response.
2) When $\mathcal{CS}^*$ chooses the challenge message $k_{0,1}, k_{1,1}$, $\mathcal{M}$ chooses a uniform bit $b \in \{0, 1\}$, queries $\mathcal{O}(k_{b,1})$, obtains a response, and returns the strings to $\mathcal{CS}^*$.
3) Continue answering queries of $\mathcal{CS}^*$ as before until $\mathcal{CS}^*$ outputs b'. Outputs 1 if $b = b'$, and 0 otherwise.

If $\mathcal{O}$ returns strings generated by $\mathrm{F}(\cdot)$,

$$\Pr[\mathcal{M}^{ck_1,\ldots,\tilde{ck}_n}(1^\ell, n, PW) \Rightarrow 1] = \Pr[\mathrm{G}_0(1^\ell, n, PW) = 1]. \quad (2)$$

If $\mathcal{O}$ returns strings generated by $f(\cdot)$,

$$\Pr[\mathcal{M}^{rs_1,\ldots,rs_n}(1^\ell, n, PW) \Rightarrow 1] = \Pr[\mathrm{G}_1(1^\ell, n, PW) = 1]. \quad (3)$$

The possibility for $\mathcal{M}$ to distinguish whether a string output by the pseudorandom function or the truly random function is negligible. Therefore,

$$|\Pr[\mathcal{M}^{ck_1,\ldots,\tilde{ck}_n}(1^\ell, n, PW) \Rightarrow 1] - \Pr[\mathcal{M}^{rs_1,\ldots,rs_n}(1^\ell, n, PW) \Rightarrow 1]| \leq \mathrm{negl}(n). \quad (4)$$

Furthermore, in G_1, $rs_1, rs_2, \ldots, rs_n$ are output by a random function, $\mathcal{CS}^*$ learns nothing about $k_{b,1}$ from them. Thus,

$$\Pr[\mathrm{G}_1(1^\ell, n, PW) = 1] = \frac{1}{2} + \frac{p(\ell)}{2^\ell},$$

where $p(\ell)/2^\ell$ is essentially the possibility that $\mathcal{CS}^*$ guesses spw correctly.

Combined with Eqs. (2), (3), and (4), we have

$$\begin{aligned}\Pr[\mathrm{G}_0(1^\ell, n, PW) = 1] &\leq \Pr[\mathrm{G}_1(1^\ell, n, PW) = 1] + \mathrm{negl}(\ell) \\ &= \frac{1}{2} + \frac{p(\ell)}{2^\ell} + \mathrm{negl}(\ell).\end{aligned}$$

This concludes the proof. □

Note that Theorem 1 implies the security against DGA. In terms of online DGA, the possibility of success is $\frac{q(\ell)}{|PW|}$, where $q(\ell)$ can be limited to a small constant by limiting the times that the adversary fails to guess pw correctly [7], [29], such that the possibility is negligible. In terms of offline DGA, the possibility of success is $\frac{p(\ell)}{2^\ell}$, which is also a negligible value. Therefore, HealthFort is able to resist against DGA.

6.2 Provenance Security

Resistance Against Forgery and Deletion. In HealthFort, when a doctor colludes with the cloud server to forge a provenance record, the doctor needs to store its information on Ethereum with the corresponding patient's permission. Without the permission, the provenance record cannot be integrated into the blockchain and cannot be valid neither. If a provenance record is deleted, its corresponding information stored on Ethereum needs to be deleted. However, due to the tamper-proof property of the blockchain, launching attacks on data stored on Ethereum requires more than half of the computational power of the entire network, which may not be even cost worthy.

Identity Privacy Preservation. A doctor's identity in a diagnosis is privacy information and is included in several provenance records. As provenance records are publicly stored, a malicious adversary may extract the information about the doctor's identity from provenance records by following strategies:

1) The adversary obtains the real identity of the doctor in a diagnosis.
2) The adversary determines which provenance records are generated by the same doctor.

For the first strategy, instead of the doctor's real identity, a pseudorandom identity is included in the provenance records. The pseudorandom identity is generated by the hospital by encrypting the real identity with a random treatment key. The adversary cannot retrieve the real identity of the doctor without the treatment key. For the second strategy, each time a pseudorandom identity is generated, the hospital randomly chooses a treatment key as the encryption key. In such case, among different provenance records, which includes the pseudorandom identity corresponding to the same doctor cannot be determined.

Non-Repudiation. In HealthFort, if a doctor repudiates a provenance record generated by her/him, there are two possible strategies:

1) The doctor claims that the pseudonym included in the provenance record does not belong to her/him.
2) The doctor claims that others use her/his pseudonym to generate and publish the provenance record.

For the first strategy, the hospital generates the pseudonym that is derived from the doctor's real identity. The hospital can prove that the pseudonym belongs to the doctor by extracting the identity from the provenance record. For strategy 2, before the provenance record is outsourced, the creator of the provenance record sends it to the hospital, the hospital verifies the provenance record, where the pseudonym is included. If the pseudonym does not match the real identity of the creator, the hospital rejects to sign on the provenance record. Without the signature of the hospital on it, the cloud server will not accept the provenance record. Therefore, both strategies are infeasible, the doctor cannot repudiate the provenance record generated by her/him. Due to the activities recorded in the provenance record, the doctor cannot deny that she/he generated some EHRs for

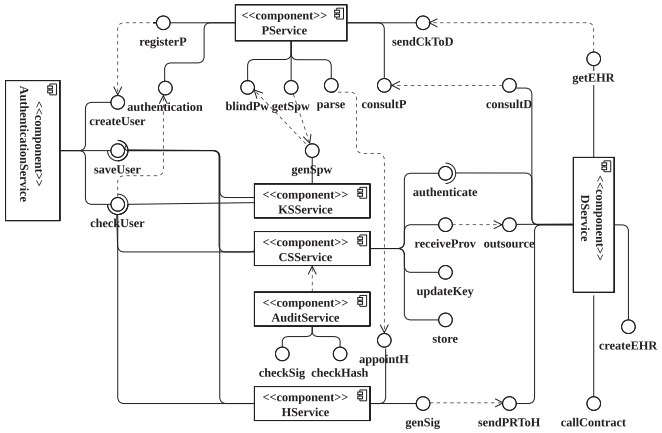


Fig. 13. The component diagram of the HealthFort prototype.

the patient either. With identity privacy preservation and non-repudiation, in other words, HealthFort satisfies conditional identity privacy preservation.

7 Implementation and Evaluation

We implement a HealthFort prototype in Java, JavaScript, and Solidity in more than 32000 lines of code. Specifically, Java is used to implement the back-end of HealthFort, JavaScript is utilized to implement the front-end of HealthFort, and Solidity is used to implement the smart contract. A component diagram of the prototype is shown in Fig. 13 and our source code is available on the following address: https://github.com/sssshai/HealthFort

In the implementation, the security level is chosen to be 80 bits, where the underlying elliptic curve is an alt_bn128 curve whose base field size is 128 bits. The cryptography algorithms are implemented based on JPBC library[5], h and $\hbar$ are implemented by using SHA256, the encryption algorithm $\mathrm{E}(\cdot)$ is chosen to be AES-256 [57], the signature algorithm $\mathrm{Sig}(\cdot)$ is chosen to be BLS [58].

We conduct experiments on a laptop with macOS, an Intel Core i5 CPU, and 16 GB DDR4X of RAM. We consider the network latency and transaction synchronous under the public blockchain network, evaluate the performance of our scheme in the Ethereum public test network *Ropsten* using *MetaMask* as the wallet App.

7.1 Implementation

Now we introduce the methods for each algorithm interpreted in Section 5, a detailed list of the methods is provided in Table 3.

Registration. We implement `blindPW`$(\cdot)$ for a patient to blind the password. Key servers invoke `genSpw`$(\cdot)$ to generate and share a servers-hardened key, which outputs σ_i^* and Q_i. Then the patient can obtain a servers-hardened password $spw_{\mathcal{P}}$ invoking `getSpw`$(\cdot)$, and further compute $au_{\mathcal{KS}}, au_{\mathcal{CS}}, au_{\mathcal{H}}$, and $csk_{\mathcal{P}}$ to register invoking `registerP`$(\cdot)$. After the key servers, the cloud server, and the hospital invoke `saveUser`$(\cdot)$ to store the patient's information, the registration is completed.

Appointment. The patient invokes `authenticate`$(\cdot)$ to authenticate with the hospital, and the hospital invokes `checkUser`$(\cdot)$ to validate the validity of the patient's identity. After authentication, the hospital invokes `appointH`$(\cdot)$ to designate a doctor for the patient and to send appointment information to the patient. On the patient side, `parse`$(\cdot)$ is used to parse the appointment information sent from the hospital.

Consultation. The patient invokes `authenticate`$(\cdot)$ to authenticate with the cloud server and `consultP`$(\cdot)$ to consult the doctor. After validating the validity of $au_{\mathcal{CS}}$, the cloud server allows the patient to download $csk_{\mathcal{P}}$ and $\tilde{ck}_y^{(De)}$. The doctor invokes `consultD`$(\cdot)$ to validate the validity of a permission sent from the patient, generate an encryption key of the EHR generated in the current consultation and encrypt the encryption key. If the doctor needs to view the patient's previous EHRs, the patient sends $ck_y^{(De)}$ to the doctor by invoking `sendCkToD`$(\cdot)$. Then the doctor can use `getEHR`$(\cdot)$ to compute $k_y^{(De)}$ and further obtain the plaintexts of previous EHRs. After the consultation, the doctor creates an new EHR for the patient by `createEHR`$(\cdot)$.

ProvGen. We implement `sendPRToH`$(\cdot)$ and `genSig`$(\cdot)$ for this algorithm. After generating a provenance record, the doctor invokes `sendPRToH`$(\cdot)$ to request a signature of the hospital on the provenance record. The hospital invokes `genSig`$(\cdot)$ to compute a signature on the provenance record.

Store. The doctor sends a transaction to the address of the smart contract by `callContract`$(\cdot)$ to store the information of the provenance record on Ethereum. After the transaction is recorded into Ethereum, the doctor uses `outsource`$(\cdot)$ to outsource the ciphertext of the generated EHR, the corresponding provenance record, and the signature of the hospital on the provenance record to the cloud server. The patient uses `updKey`$(\cdot)$ to update the encryption keys of EHRs. The cloud server invokes `receiveProv`$(\cdot)$ to store the ciphertext of generated EHR, the corresponding provenance record, the signature of the hospital on the provenance record, and the ciphertexts of updated keys.

Audit. In this algorithm, an audit can invoke `checkSig`$(\cdot)$ to validate the validity of the signature of the hospital on the latest provenance record, and invoke `checkHash`$(\cdot)$ to extract the information from Ethereum to verify the integrity and timeliness of provenance records.

HealthFort employs a set of independent key servers to harden passwords. It seems that the patients have to bear the additional costs to employ them. In the practical deployment of HealthFort, the hospitals can serve as the key servers due to the following observations. In reality, patients always go to the hospitals in a fixed area, which means that they always interact with fixed hospitals in most cases. To meet the related regulations, multiple hospitals always utilize same cloud server to manage patients' data and even share a patient's data with each other (under the patient's permission). The hospitals are independent of each other and are not fully trusted by patients, this essentially follows the same threat model as the key servers in HealthFort. As such, patients do not need to bear additional costs to employ independent key servers.

5. The Java Pairing Based Cryptography Library (JPBC). http://gas.dia.unisa.it/projects/jpbc/#.Yg9TnC-KFqs

TABLE 3
Details of Methods Implemented in HealthFort

Algorithm	Entity	Method
Registration	Patient	$pw^*_{\mathcal{P}} \leftarrow \texttt{blindPW}(pw_{\mathcal{P}})$ $spw_{\mathcal{P}} \leftarrow \texttt{getSPW}(\sigma^*[\,], pw^*_{\mathcal{P}}, Q[\,])$ $au_{\mathcal{KS}}, au_{\mathcal{CS}}, au_{\mathcal{H}}, csk_{\mathcal{P}} \leftarrow \texttt{registerP}(spw_{\mathcal{P}}, ID_{\mathcal{KS}}, ID_{\mathcal{CS}}, ID_{\mathcal{H}}, sk_{\mathcal{P}})$
	Key servers	$\sigma^*[\,], Q[\,] \leftarrow \texttt{genSpw}(pw^*_{\mathcal{P}})$ $\texttt{saveUser}(ID_{\mathcal{P}}, au_{\mathcal{KS}}[\,])$
	Cloud server	$\texttt{saveUser}(ID_{\mathcal{CS}}, au_{\mathcal{CS}}, csk_{\mathcal{P}})$
	Hospital	$\texttt{saveUser}(ID_{\mathcal{P}}, au_{\mathcal{H}})$
Appointment	Patient	$au_{\mathcal{H}} \leftarrow \texttt{authenticate}(ID_{\mathcal{P}}, pw_{\mathcal{P}})$ $ID_{\mathcal{D}}, PID_{\mathcal{D}}, TP_{\mathcal{D}}, Aux_{\mathcal{D}} \leftarrow \texttt{parse}(aptInf)$
	Hospital	$\texttt{checkUser}(ID_{\mathcal{P}}, au_{\mathcal{H}})$ $aptInf \leftarrow \texttt{appointH}(ID_{\mathcal{P}})$
Consultation	Patient	$au_{\mathcal{CS}} \leftarrow \texttt{authenticate}(ID_{\mathcal{P}}, pw_{\mathcal{P}})$ $k^{(De)}_{y+1}, per_{\mathcal{D}} \leftarrow \texttt{consultP}(csk_{\mathcal{P}}, \tilde{ck}^{(De)}_{y})$ $ck^{(De)}_{y} \leftarrow \texttt{sendCkToD}(ID_{\mathcal{P}}, \tilde{ck}^{(De)}_{y})$
	Cloud server	$\texttt{checkUser}(ID_{\mathcal{P}}, au_{\mathcal{CS}})$
	Doctor	$ek^{(De)}_{y+1} \leftarrow \texttt{consultD}(per_{\mathcal{D}})$ $EHR^{(De)}_1, EHR^{(De)}_2, ..., EHR^{(De)}_y \leftarrow \texttt{getEHR}(per_{\mathcal{D}}, ck^{(De)}_{y})$ $EHR^{(De)}_{y+1} \leftarrow \texttt{createEHR}(ID_{\mathcal{P}})$
ProvGen	Doctor	$\texttt{sendPRToH}(\sigma^{(PR)}_{\iota-1}, PR_{\iota})$
	Hospital	$\sigma^{(PR)}_{\iota} \leftarrow \texttt{genSig}(PR_{\iota})$
Store	Doctor	$\texttt{callContract}(per_{\mathcal{D}}, \sigma^{(PR)}_{\iota}, PR_{\iota})$
	Patient	$\texttt{updKey}(EHR^{(De)}_{y+1}, k^{(De)}_{y+1})$
	Cloud server	$\texttt{receiveProv}(per_{\mathcal{D}}, EHR^{(De)}_{y+1}, \sigma^{(PR)}_{\iota}, PR_{\iota}, ck^{(De)}_{y}, \tilde{ck}^{(De)}_{y})$
Audit	Auditor	$\texttt{checkSig}(ID_{\mathcal{P}})$ $\texttt{checkHash}(Add, ID_{\mathcal{P}})$

7.2 Comparison and Evaluation

We compare HealthFort with existing works in terms of security and functionality. The comparison with eHealth systems, e.g., JMEK09 [14], SAGE [59], SMR10 [32], HealthDep [12], and ASBKS [38] is shown in Table 4. As regards efficiency, we compare the computation costs of the provenance mechanism in HealthFort with existing provenance mechanisms, e.g., LLLS10 [22], CCHZD12 [23], ProvChain [24], ZLX18 [26] and SECProv [21]. The comparison with provenance mechanisms is provided in Table 5. The notations used in this two tables are described in Table 6.

TABLE 4
Comparison With eHealth Systems

	JMEK09 [14]	SAGE [59]	SMR10 [32]	HealthDep [12]	ASBKS [38]	HealthFort
Underlying encryption mechanism for EHR confidentiality	Public-key	Symmetric-key	Public-key	Symmetric-key	Public-key	Symmetric-key
Portability	N	N	N	N	N	Y
Conditional forward transparency	Y	N	Y	N	Y	Y
No single point of failure	Y	N	N	Y	N	Y
Backward security	N	N	Y	Y	Y	Y

TABLE 5
Comparison With Provenance Mechanisms

	LLLS10 [22]	CCHZD12 [23]	ProvChain [24]	ZLX18 [26]	SECProv [21]	HealthFort
No single point of failure	N	N	Y	Y	Y	Y
Compatible with eHealth systems	N	N	N	N	N	Y
Conditional identity privacy preservation	Y	$\perp$	$\perp$	$\perp$	$\perp$	Y
Data structure of provenance records	N	N	Y	N	Y	Y
Computation costs on storing a provenance record	$16\text{Exp}_G + 11\text{Mul}_G$ $+6\text{Pair} + 3\text{Add}_{Z_{\mathbb{N}}^*}$ $+2\text{H}_{Z_{\mathbb{N}}^*}$	$3\text{Add}_{Z_p} + 3\text{Mul}_{Z_p}$ $+2\text{H}_{Z_p} + 5\text{Mul}_G$ $+14\text{Exp}_G + 2\text{H}_G$ $+\text{H}_{G^2} + 6\text{Pair}$	$\perp$	$3\text{H}_{Z_p^*} + 2\text{Pair}$ $+3\text{E} + 2\text{H}_G$ $+7\text{Mul}_G$	$2\text{Exp}_G + 3\text{H}_G$ $+\text{Mul}_G + 2\text{Pair}$ $+3\text{H}_{Z_p}$	$6\text{Pair} + 4\text{H}_G +$ $2\text{H}_{Z_p} + \text{Mul}_G$

Now we present the evaluation results of the HealthFort prototype, focusing on three aspects: 1) Computational costs; 2) Communication costs; 3) Practicality.

1) *Computational costs.* On the patient side, we notice that the computation costs change with the total number n and the threshold number t of key servers. The patient needs to compute the servers-hardened password spw using the t signatures sent from key servers. Therefore, we evaluate the computation costs of the whole procedure on the patient side with different (t, n), and the results of experiments are shown in Fig. 14a. According to the experiment results, we can observe that when $t = 7$ and $n = 20$, the computation delay of the whole procedure, including registering, is no more than 1.2 seconds, which is acceptable for patients. In addition, we evaluate the computation costs of the patient with different consultation times, and the results of experiments are shown in Fig. 14b, where we fix $n = 20$. According to the experiment results, we observe that the computation delay for a registered patient to consult a doctor is about 0.16 seconds.

On the key server side, the main computation costs are (t, n)-threshold secret generating and sharing, which are one-time costs. Once a secret has been generated and shared between key servers, each of them only needs to generate a signature on the blinded password when the patient requests a servers-hardened password. We evaluate the computation delay of each key server with different (t, n), the results are shown in Fig. 14c. According to the results of the experiments, we observe that when $t = 7$ and $n = 20$, the computation delay for a key server to assist a patient with a consultation is about 6.5 seconds.

On the doctor side, the computation delay is shown in Fig. 14d, being consulted by 100 patients only takes about 15 seconds. In reality, there are at most 80 patients diagnosed by one doctor per day.

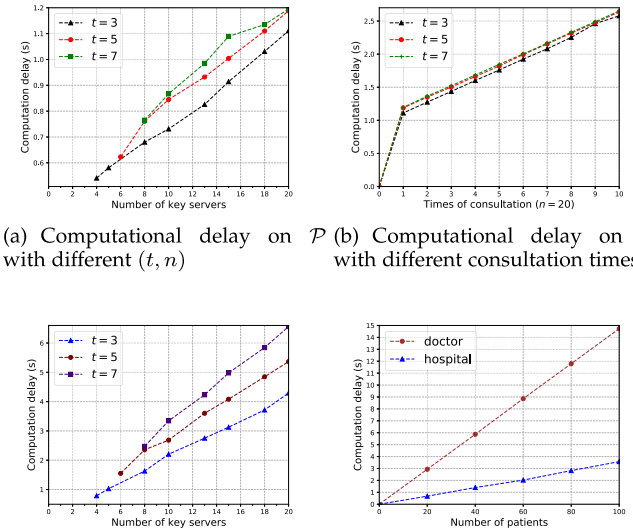
On the hospital side, the computation delay is mainly caused by generating a signature on a provenance record. We conduct experiments on the computation delay of the hospital with different patients, which is shown in Fig. 14d. According to the results of experiments, we can observe that providing service for 100 patients takes no more than 4 seconds.

2) *Communication costs.* Patients interact with all key servers to request the service of password hardening. On the patient side, the communication costs increase with the increase of the number of key servers n. We conduct experiments on the communication costs of a patient, the results are shown in Fig. 15a. We can observe that when $n = 20$, the communication costs for a patient to complete a consultation are about 12 KB.

When a patient requests password hardening, each key server communicates with others to generate server-side

TABLE 6
Notations Used in Table 4 and Table 5

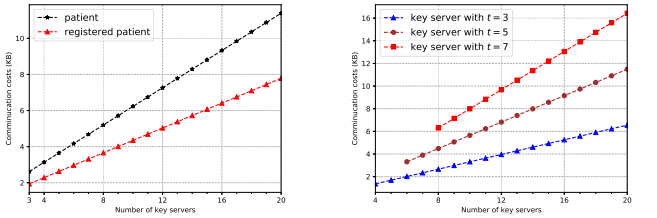
Notation	Description
Y	The system fulfills the requirement.
N	The system does not fulfill the requirement.
$\perp$	The system does not focus on the requirement.
$\mathbb{N}$	The RSA module.
Add_S	Addition in the set S.
Mul_S	Multiplication in the set S.
Exp_S	Exponentiation in the set S.
H_S	Hashing a value into the set S.
Pair	Bilinear pairing.
E	Symmetrical encryption.



(a) Computational delay on $\mathcal{P}$ with different (t, n) (b) Computational delay on $\mathcal{P}$ with different consultation times

(c) Computational delay on $\mathcal{KS}_i$ (d) Computational delay on $\mathcal{D}$, $\mathcal{H}$

Fig. 14. Computational delay.



(a) Communication costs on $\mathcal{P}$ (b) Communication costs on $\mathcal{KS}_i$

Fig. 15. Communication costs.

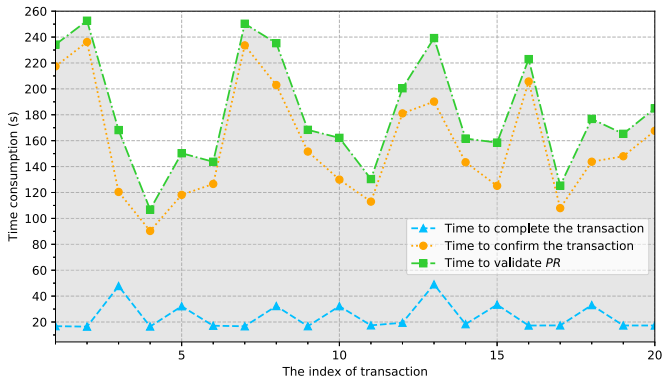


Fig. 16. Evaluation of timeliness.

secret distributed into t parts. Therefore, on the key server side, the communication costs change with n and t. We conduct experiments on communication costs on the key server, the results are shown in Fig. 15b. Once the key servers complete the secret sharing, the service for a patient only takes each of them about 0.3 KB every time.

On the doctor side, the communication costs are mainly sending the provenance record to the hospital to request a signature, calling the smart contract, and outsourcing the new EHR and the corresponding provenance to the cloud server. The communication costs on the doctor side are within 200 KB when he/she diagnoses 100 patients.

3) *Practicality*. The practicality of HealthFort is evaluated in three aspects. First, the time consumption that a provenance record PR to be valid after being generated, which mainly consists of two aspects. The first one is the time to record the transaction to the blockchain; The second one is the confirmation time for the transaction. We conduct 20 transactions by invoking the smart contract that we deployed in our system, and the evaluation results are shown in Fig. 16. The time consumption of validating a provenance record is around 2.8 minutes. Since the blockchain system would dynamically adjust the difficulty, generally the time consumption would not exceed 300 seconds.

Second, the monetary costs to deploy a smart contract and to invoke the storage function of the smart contract are measured. On *Ropsten*, deploying the smart contract costs 783379 gas and invoking the storage function once costs 298390 gas. As of Sep. 2021, deploying the smart contract requires about 0.0423025 ETH, and storing the information about a provenance record requires about 0.0161131 ETH. This can be acceptable to patients in respect of the value of the provenance record protected by HealthFort.

Third, in HealthFort, the security of the provenance relies on a blockchain (i.e., the Ethereum blockchain in this paper), the requirement for the blockchain is to perpetually store hash values of the provenance information. As tamper-proof is a basic feature for blockchain systems, HealthFort is compatible with other secure blockchains.

From the results of the evaluation, it is demonstrated that HealthFort is efficient and practical in terms of computation, communication, and monetary costs.

8 Conclusion

In this paper, we have proposed a servers-aided password-based subsequent-key-locked encryption mechanism to ensure EHRs' confidentiality, and a blockchain-based secure data provenance mechanism for eHealth systems. Based on the two mechanisms, we have developed HealthFort, to provide a secure storage services for EHRs with efficient provenance. We have conducted a formal analysis and comprehensive evaluation on HealthFort to demonstrate that it is secure, efficient, and practical for implementation and development. For the future work, we will design a secure data provenance mechanism for cyber supply chain, e-science, and database, etc.

References

[1] S. Li, Y. Zhang, C. Xu, N. Cheng, Z. Liu, and X. Shen, "BESURE: Blockchain-based cloud-assisted ehealth system with secure data provenance," in *Proc. IEEE/ACM 29th Int. Symp. Qual. Serv.*, 2021, pp. 1–6.

[2] L. Guo, C. Zhang, J. Sun, and Y. Fang, "A privacy-preserving attribute-based authentication system for mobile health networks," *IEEE Trans. Mobile Comput.*, vol. 13, no. 9, pp. 1927–1941, Sep. 2014.

[3] X. Shen et al., "Blockchain for transparent data management towards 6G," *Engineering*, vol. 8, pp. 74–85, 2022.

[4] M.-H. Chen, M. Dong, and B. Liang, "Resource sharing of a computing access point for multi-user mobile cloud offloading with delay constraints," *IEEE Trans. Mobile Comput.*, vol. 17, no. 12, pp. 2868–2881, Dec. 2018.

[5] S. Moulik, S. Misra, and A. Gaurav, "Cost-effective mapping between wireless body area networks and cloud service providers based on multi-stage bargaining," *IEEE Trans. Mobile Comput.*, vol. 16, no. 6, pp. 1573–1586, Jun. 2016.

[6] X. Li, H. Wang, C. Chen, and J. Grundy, "An empirical study on how well do COVID-19 information dashboards service user information needs," *IEEE Trans. Serv. Comput.*, vol. 15, no. 3, pp. 1178–1192, May/Jun. 2022.

[7] Y. Zhang, C. Xu, N. Cheng, and X. Shen, "Secure password-protected encryption key for deduplicated cloud storage systems," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 4, pp. 2789–2806, Jul./Aug. 2022.

[8] J. Liang, Z. Qin, S. Xiao, L. Ou, and X. Lin, "Efficient and secure decision tree classification for cloud-assisted online diagnosis services," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 4, pp. 1632–1644, Apr. 2019.

[9] J. Liu, C. Zhang, K. Xue, and Y. Fang, "Privacy preservation in multi-cloud secure data fusion for infectious-disease analysis," *IEEE Trans. Mobile Comput.*, early access, Jan. 25, 2022, doi: 10.1109/TMC.2022.3145745.

[10] Y. Wang, Q. Wu, B. Qin, W. Shi, R. H. Deng, and J. Hu, "Identity-based data outsourcing with comprehensive auditing in clouds," *IEEE Trans. Inf. Forensics Secur.*, vol. 12, no. 4, pp. 940–952, Apr. 2017.

[11] X. Li, S. Liu, R. Lu, M. K. Khan, K. Gu, and X. Zhang, "An efficient privacy-preserving public auditing protocol for cloud-based medical storage system," *IEEE J. Biomed. Health Inform.*, vol. 26, no. 5, pp. 2020–2031, May 2022.

[12] Y. Zhang, C. Xu, H. Li, K. Yang, J. Zhou, and X. Lin, "HealthDep: An efficient and secure deduplication scheme for cloud-assisted ehealth systems," *IEEE Trans. Ind. Informat.*, vol. 14, no. 9, pp. 4101–4112, Sep. 2018.

[13] J. Wei, X. Chen, J. Wang, X. Hu, and J. Ma, "Enabling (end-to-end) encrypted cloud emails with practical forward secrecy," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 4, pp. 2318–2332, Jul./Aug. 2022.

[14] J. Benaloh, M. Chase, E. Horvitz, and K. Lauter, "Patient controlled encryption: Ensuring privacy of electronic medical records," in *Proc. ACM Workshop Cloud Comput. Secur.*, 2009, pp. 103–114.

[15] M. Shen, J. Zhang, L. Zhu, K. Xu, X. Du, and Y. Liu, "Encrypted traffic classification of decentralized applications on ethereum using feature fusion," in *Proc. Int. Symp. Qual. Serv.*, 2019, pp. 47–52.

[16] W. Lee and C. Lee, "A cryptographic key management solution for hipaa privacy/security regulations," *IEEE Trans. Informat. Technol. Biomed.*, vol. 12, no. 1, pp. 34–41, Jan. 2008.

[17] K.-K. Muniswamy-Reddy, D. A. Holland, U. Braun, and M. I. Seltzer, "Provenance-aware storage systems," in *Proc. Annu. Conf. USENIX Annu. Tech. Conf.*, 2006, pp. 43–56.

[18] K.-K. Muniswamy-Reddy and M. Seltzer, "Provenance as first class cloud data," *ACM SIGOPS Operating Syst. Rev.*, vol. 43, no. 4, pp. 11–16, 2010.

[19] K.-K. Muniswamy-Reddy, P. Macko, and M. I. Seltzer, "Provenance for the cloud," in *Proc. FAST*, 2010, pp. 15–14.

[20] M. R. Asghar, M. Ion, G. Russello, and B. Crispo, "Securing data provenance in the cloud," in *Proc. Int. Workshop Open Problems Netw. Secur.*, 2012, pp. 145–160.

[21] S. Zawoad, R. Hasan, and K. Islam, "SECProv: Trustworthy and efficient provenance management in the cloud," in *Proc. IEEE Conf. Comput. Commun.*, 2018, pp. 1241–1249.

[22] R. Lu, X. Lin, X. Liang, and X. Shen, "Secure provenance: The essential of bread and butter of data forensics in cloud computing," in *Proc. 5th ACM Symp. Inf. Comput. Commun. Secur.*, 2010, pp. 282–292.

[23] S. S. Chow, C. Chu, X. Huang, J. Zhou, and R. H. Deng, "Dynamic secure cloud storage with provenance," in *Proc. Cryptography Secur. From Theory Appl.*, 2012, pp. 442–464.

[24] X. Liang, S. Shetty, D. Tosh, C. Kamhoua, K. Kwiat, and L. Njilla, "Provchain: A blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability," in *Proc. 17th IEEE/ACM Int. Symp. Cluster Cloud Grid Comput.*, 2017, pp. 468–477.

[25] R. Neisse, G. Steri, and I. Nai-Fovino, "A blockchain-based approach for data accountability and provenance tracking," in *Proc. 12th Int. Conf. Availability Rel. Secur.*, 2017, pp. 1–10.

[26] Y. Zhang, X. Lin, and C. Xu, "Blockchain-based secure data provenance for cloud storage," in *Proc. Int. Conf. Inf. Commun. Secur.*, 2018, pp. 3–19.

[27] N. Tyagi, I. Miers, and T. Ristenpart, "Traceback for end-to-end encrypted messaging," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2019, pp. 413–430.

[28] S. Agrawal, P. Miao, P. Mohassel, and P. Mukherjee, "Pasta: Password-based threshold authentication," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2018, pp. 2042–2059.

[29] Y. Zhang, C. Xu, H. Li, K. Yang, N. Cheng, and X. Shen, "Protect: Efficient password-based threshold single-sign-on authentication for mobile users against perpetual leakage," *IEEE Trans. Mobile Comput.*, vol. 20, no. 6, pp. 2297–2312, Jun. 2020.

[30] S. Shang, X. Li, R. Lu, J. Niu, X. Zhang, and M. Guizani, "A privacy-preserving multi-dimensional range query scheme for edge-supported industrial IoT," *IEEE Internet Things J.*, vol. 9, no. 16, pp. 15285–15296, Aug. 2022.

[31] H. Yue, L. Guo, R. Li, H. Asaeda, and Y. Fang, "Dataclouds: Enabling community-based data-centric services over the Internet of Things," *IEEE Internet Things J.*, vol. 1, no. 5, pp. 472–482, Oct. 2014.

[32] S. Narayan, M. Gagné, and R. Safavi-Naini, "Privacy preserving EHR system using attribute-based infrastructure," in *Proc. ACM Workshop Cloud Comput. Secur. Workshop*, 2010, pp. 47–52.

[33] K. Zhang, K. Yang, X. Liang, Z. Su, X. Shen, and H. H. Luo, "Security and privacy for mobile healthcare networks: From a quality of protection perspective," *IEEE Wireless Commun.*, vol. 22, no. 4, pp. 104–112, Apr. 2015.

[34] A. Act, "Health insurance portability and accountability act of 1996," *Public Law*, vol. 104, 1996, Art. no. 191.

[35] J. Sun, X. Zhu, C. Zhang, and Y. Fang, "HCPP: Cryptography based secure EHR system for patient privacy and emergency healthcare," in *Proc. 31st Int. Conf. Distrib. Comput. Syst.*, 2011, pp. 373–382.

[36] Y. Chen, J. Lu, and J. Jan, "A secure EHR system based on hybrid clouds," *J. Med. Syst.*, vol. 36, no. 5, pp. 3375–3384, 2012.

[37] X. Zhang, C. Xu, H. Wang, Y. Zhang, and S. Wang, "FS-PEKS: Lattice-based forward secure public-key encryption with keyword search for cloud-assisted industrial internet of things," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 3, pp. 1019–1032, May-Jun. 2019.

[38] L. Xu et al., "ASBKS: Towards attribute set based keyword search over encrypted personal health records," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 6, pp. 2941–2952, Nov.-Dec. 2021.

[39] H. Huang, X. Sun, F. Xiao, P. Zhu, and W. Wang, "Blockchain-based ehealth system for auditable EHRs manipulation in cloud environments," *J. Parallel Distrib. Comput.*, vol. 148, pp. 46–57, 2021.

[40] J. Wei, X. Chen, X. Huang, X. Hu, and W. Susilo, "RS-HABE: Revocable-storage and hierarchical attribute-based access scheme for secure sharing of E-health records in public cloud," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 5, pp. 2301–2315, Sep./Oct. 2021.

[41] J. Wei, X. Chen, J. Ma, X. Hu, and K. Ren, "Communication-efficient and fine-grained forward-secure asynchronous messaging," *IEEE/ACM Trans. Netw.*, vol. 29, no. 5, pp. 2242–2253, May 2021.

[42] M. Li, S. Yu, Y. Zheng, K. Ren, and W. Lou, "Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 1, pp. 131–143, Jan. 2012.

[43] Y. Miao et al., "Privacy-preserving attribute-based keyword search in shared multi-owner setting," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 3, pp. 1080–1094, Mar. 2019.

[44] J. Li, X. Chen, M. Li, J. Li, P. P. Lee, and W. Lou, "Secure deduplication with efficient and reliable convergent key management," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 6, pp. 1615–1625, Jun. 2013.

[45] B. Lee, A. Awad, and M. Awad, "Towards secure provenance in the cloud: A survey," in *Proc. IEEE/ACM 8th Int. Conf. Utility Cloud Comput.*, 2015, pp. 577–582.

[46] C. A. Lynch, "When documents deceive: Trust and provenance as new factors for information retrieval in a tangled web," *J. Amer. Soc. Informat. Sci. Technol.*, vol. 52, no. 1, pp. 12–17, 2001.

[47] R. Hasan, R. Sion, and M. Winslett, "Introducing secure provenance: Problems and challenges," in *Proc. ACM Workshop Storage Secur. Survivability*, 2007, pp. 13–18.

[48] O. Q. Zhang, M. Kirchberg, R. K. Ko, and B. S. Lee, "How to track your data: The case for cloud computing provenance," in *Proc. IEEE 3rd Int. Conf. Cloud Comput. Technol. Sci.*, 2011, pp. 446–453.

[49] R. K. Ko and M. A. Will, "Progger: An efficient, tamper-evident kernel-space logger for cloud data provenance tracking," in *Proc. IEEE 7th Int. Conf. Cloud Comput.*, 2014, pp. 881–889.

[50] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized Bus. Rev.*, p. 21260, 2008.

[51] G. Wood et al., "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, vol. 151, no. 2014, pp. 1–32, 2014.

[52] Y. Zhang, C. Xu, N. Cheng, H. Li, H. Yang, and X. Shen, "Chronos +: An accurate blockchain-based time-stamping scheme for cloud storage," *IEEE Trans. Serv. Comput.*, vol. 13, no. 2, pp. 216–229, Feb. 2019.

[53] N. Szabo, "Smart contracts: Building blocks for digital markets," *J. Transhumanist Thought*, vol. 18, no. 2, pp. 50–53, 1996.

[54] D. Wang, Z. Zhang, P. Wang, J. Yan, and X. Huang, "Targeted online password guessing: An underestimated threat," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2016, pp. 1242–1254.

[55] T. Lebo et al., "PROV-O: The prov ontology," *W3C Recommendation*, vol. 30, 2013.

[56] S. Jarecki, A. Kiayias, H. Krawczyk, and J. Xu, "TOPPSS: Cost-minimal password-protected secret sharing based on threshold OPRF," in *Proc. Int. Conf. Appl. Cryptography Netw. Secur.*, 2017, pp. 39–58.

[57] M. J. Dworkin et al., "Advanced encryption standard (AES)," 2001. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf

[58] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," in *Proc. Int. Conf. Theory Appl. Cryptology Inf. Secur.*, 2001, pp. 514–532.

[59] X. Lin, R. Lu, X. Shen, Y. Nemoto, and N. Kato, "SAGE: A strong privacy-preserving scheme against global eavesdropping for ehealth systems," *IEEE J. Sel. Areas Commun.*, vol. 27, no. 4, pp. 365–378, Apr. 2009.

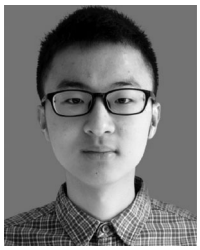
Shiyu Li received the BSc degree from the University of Electronic Science Technology of China (UESTC), in 2021. She is currently working toward the master degree with the School of Computer Science and Engineering, University of Electronic Science Technology of China. Her research interests are applied cryptography, data security, and blockchain technology.

Yuan Zhang (Member, IEEE) received the BSc degree and PhD degree from the University of Electronic Science Technology of China (UESTC), in 2013 and 2019, respectively. He is currently working toward the PhD degree from 2017 to 2019 in BBCR Lab, Department of ECE, University of Waterloo, Canada. He is currently an associate professor with the School of Computer Science and Engineering with UESTC. His research interests are applied cryptography, data security, and blockchain technology.

Chunxiang Xu (Member, IEEE) received the BSc and MSc degrees in applied mathematics from Xidian University, Xi'an, China, in 1985 and 2004, respectively, and the PhD degree in cryptography from Xidian University, in 2004. She is currently a professor with the Center for Cyber Security, School of Computer Science and Engineering, UESTC. Her research interests include information security, cloud computing security, and cryptography.

Nan Cheng (Member, IEEE) received the BE and MS degrees from Tongji University, Shanghai, China, in 2009 and 2012, respectively, and the PhD degree from the University of Waterloo, Waterloo, ON, Canada, in 2016. He is currently a professor with the School of Telecommunication, Xidian University. His current research focuses on big data in vehicular networks and self-driving system.

Zhi Liu (Member, IEEE) received the PhD degree in informatics from National Institute of Informatics. He is currently an Associate Professor with the University of Electro-Communications. His research interest includes video network transmission and mobile edge computing. He is now an editorial board member of Springer wireless networks and IEEE Open Journal of the Computer Society.



Yicong Du received the BSc and MSc degree from the University of Electronic Science Technology of China, in 2018 and 2021, respectively. He is currently working toward the PhD degree with the School of Computer Science and Engineering, University of Electronic Science and Technology of China. His research interests are applied cryptography and blockchain technology, with the current focus on physical layer security.

Xuemin Shen (Fellow, IEEE) received the PhD degree in electrical engineering from Rutgers University, New Brunswick, NJ, USA, in 1990. He is currently a University professor with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada. His research focuses on resource management in interconnected wireless/wired networks, wireless network security, social networks, smart grid, and vehicular ad hoc and sensor networks. He is a registered Professional Engineer of Ontario, Canada, an Engineering Institute of Canada Fellow, a Canadian Academy of Engineering Fellow, a Royal Society of Canada Fellow, and a Distinguished Lecturer with the IEEE Vehicular Technology Society and Communications Society. He received the R.A. Fessenden Award in 2019 from IEEE, Canada, the James Evans Avant Garde Award in 2018 from the IEEE Vehicular Technology Society, the Joseph LoCicero Award in 2015 and the Education Award in 2017 from the IEEE Communications Society. He has also received the Excellent Graduate Supervision Award in 2006 and the Outstanding Performance Award 5 times from the University of Waterloo and the Premier's Research Excellence Award (PREA) in 2003 from the Province of Ontario, Canada. He served as the Technical Program Committee Chair/Co-Chair for the IEEE Globecom'16, the IEEE Infocom'14, the IEEE VTC'10 Fall, the IEEE Globecom'07, the Symposia Chair for the IEEE ICC'10, the Tutorial Chair for the IEEE VTC'11 Spring, the Chair for the IEEE Communications Society Technical Committee on Wireless Communications, and P2P Communications and Networking. He is the Editor-in-Chief of the *IEEE Internet Of Things Journal* and the Vice President on Publications of the IEEE Communications Society.