Secure and Privacy-Preserving Sharing of Personal Health Records with Multi-Party Pre-Authorization Verification

Kheng-Leong Tan, Chi-Hung Chi and Kwok-Yan Lam, *Senior Member, IEEE*

Abstract— Wireless communications play an important role in ensuring ease of access to shared electronic health records (EHR) across health service providers which is essential and significant for prompt patients' care, especially in cases of emergency medical conditions. With the need to support anytime, anywhere access to, potentially bandwidth hungry, medical records, electronic healthcare applications will continue to benefit from advanced wireless network technologies such as 5G and beyond. With sharing, it is crucial to provide patients with security and privacy guarantees, and allow them to certain control of access to their data. Existing solutions mostly assume that patients are available to authorize requests to access their EHR, which is impractical as the patient may be unconscious. This paper proposes a secure and privacy protecting protocol whereby the patient can pre-delegate the authorization for the access of his/her EHR. Our patient(user)-centric proposal combines Self-Sovereign Identity (SSI) concepts and model with Secure Multi-party Computation (SMPC) and Threshold Cryptography (TC) to enable secure identity and authorization verification. A block cipher encryption sharing approach is adopted for the threshold SMPC which extends the AES-GCM symmetric encryption model into a full-fledged cryptographic platform. Two mechanisms are implemented for the block cipher encryption, namely XOR and Cascade, and experiments are conducted to compare them. We conclude that the XOR mechanism can scale for larger thresholds, while Cascade performed better for a lower threshold (≤ 3). This paper also performs a threat analysis of the protocol and approach, and validates its correctness and complexity. We conclude that the approach can achieve the security and privacy protection of the patient's personal EHR, as well as the autonomy of the patient to control the authorization for the access and sharing.

Keywords — Algorithm/protocol design and analysis, Cryptographic controls, Distributed Systems, Security and Privacy Protection, System architectures, integration and modeling

1 INTRODUCTION

Wireless communications play an important role in ensuring ease of access to shared electronic health records (EHR) across health service providers (HSP) which is essential and significant for prompt patients' care, especially in cases of emergency medical conditions. In medical services, communication among endpoints, including doctors, nurses, patients' medical records and monitoring data, paramedics, etc., anywhere and anytime are essentials to ensure in-time treatment of patients [1]. With advance in wireless network and the advent of 6G [2-6], this kind of communication is getting popular and widely deployed in smart healthcare and biomedical communication [7]. For example, doctors and nurses can make clinical decisions more efficiently if appropriate patient information is delivered promptly via the use of a mobile application installed on smart mobile devices, for example tablets, that have secure access to these wireless endpoints. In addition, in an emergency medical situation whereby patient requires ambulance service, patient's medical records can be made available over the wireless infrastructure to the paramedic attending to the patient and the emergency center to provide medical assistance [8]. These applications on the use of a wireless environment and wireless smart devices promote patient care and healthcare management with prompt and more efficient secure access to medical records that are encrypted and stored in a central storage [9].

Hospitals and medical clinic are HSP with healthcare professionals which provide various healthcare services at different locations. Usually, a user visits more than one healthcare professional, e.g., general practitioner, specialists, clinics, pharmacies, etc. for different needs. In a usual scenario, where users' health records that are issued by a HSP are stored locally at the provider's data system as EHR which can be accessed within its wireless network; all the management and maintenance of the data are on the provider side; only this provider is eligible to edit these records. On the aspect of availability and access to data, patients currently do not have a full, complete, let alone, comprehensive view of her/his medical history and records. Accessibility of health records by other HSP also cannot be provided on a timely basis for patients and doctors to make correct diagnoses and informed decisions. Insurance agents are unable to fully verify a client's (or claimant patient) full medical records before approving insurance claims or to facilitate checking and reviewing of complete medical information declarations made by clients during the purchase of insurance policies. Incorrect, incomplete and delayed medical information access could lead to patients paying incorrect premiums or making insurance claims that have exceeded or fall short of the actual reimbursable amounts.

For privacy and transparency issues, patients currently do not know how and to what extent have their medical records been utilised and the identities of parties that have access to their records. The level of appropriate privacy

and transparency of patients' medical records may not have been adequately guaranteed or established. Furthermore, on the fundamental matter of determining data ownership, medical records (i.e., data) are created by the HSP upon a person registering as a patient with a clinical doctor or hospital, as such, the patient and the HSP have joint ownership of the data. But authorization to access should always be conferred on the patient since s/he is the rightful data owner to decide how the data should be utilized.

In the aspect of entrusting medical records between the patient and the HSP, HSP is the data custodian. HSP is to ensure that the data is safe and secure, and to share the data only upon the patient's approval. Between or among HSPs, it is only after authorization is obtained from the patient then the data can be shared on a need-to-know basis and mutual endorsement of a data sharing agreement. Additionally, the access to data is to be allowed only over a specified time frame to ensure its usage is during the period of diagnosis or consultation. For health authorities to access data during emergencies, e.g., when the patient is unconscious, this is only to be done after the patient has delegated the authorization or prior endorsed consent/authorization to the data sharing agreement.

The use of blockchain technology ("BCT") has been advocated by research communities [10] in an attempt to overcome the challenges mentioned above and address the gaps inherently in the healthcare industry. The BCT is a decentralized database and its properties of immutability, transparency and auditability, data provenance and availability can address some of the security concerns of the EHR sharing. However, it is unable to address adequately the security requirements pertaining to data confidentiality, privacy as well as access authorization of EHR.

In current literature, various system models and, cryptographic schemes and techniques are proposed. Majority of the reviewed literature requires data owner's (e.g., the patient) approval before sharing the EHR, but they have not taken into consideration the case whereby the patient is unavailable to perform the approval. There are often scenarios, such as when the patient's health suddenly deteriorates, that require records to be made available to HSP (e.g. specialists who could be remote) or other caregivers who might not have initial access to the patient's health records. Existing authorization models follow a 'just-in-time' approach whereby the EHR data authorization must be approved by the patient when needed. This is not practical in some scenarios and moreover, the patient may not be in a state to provide this authorization when needed. Hence there is a need to develop an authorization delegation mechanism whereby the patient can pre-authorize the providers' access to his/her EHR in the event that s/he is certified as medically unfit to do so.

In regards to the issue of the control of data ownership, the notion of self-sovereign identity (SSI) has emerged in recent years. SSI is a new paradigm for online identity management [11, 12], whereby individuals and entities

can manage their digital identity and identity-related information (e.g., identifiers, attributes, credentials, behavioral traits or other personal data) by storing them locally on their own wireless devices' digital wallet (or via distributed cloud storage network) and selectively disclose and grant access to this information to authorized third parties, without the necessary need of any trusted authority or intermediary operator to validate these claims. SSI is a promising concept that could be a means of addressing the challenges of sharing and securing sensitive medical information among health service providers, as well as ensuring patients maintain sovereignty over their data.

Thus, the focus of this paper is to propose an efficient and secure protocol for the delegation of authorization [13, 14] to multi-party for the access to the EHR. The protocol facilitates the execution of the patient's pre-defined authorization to authorized parties, e.g. a panel of doctors can access the EHR when the patient is unconscious. This paper's contributions are summarized as follows:

1) Proposes and designs an authorization security protocol that enables patients, as data owners, to pre-grant selected data requesters, for instance healthcare providers, access to and share their EHR.
2) Adopts a novel approach to combine self-sovereign identity (SSI) concepts and framework with Secure Multi-party Computation (SMPC) and Threshold Cryptography (TC) to enable secure identity and authorization verification in a decentralized setup. The use of SMPC to prevent data leakage (ie, ensure security), as well as TC to ensure fault tolerance (ie, provide resilience).To the best of knowledge, this is the first research work that utilizes SSI, particularly Verifiable Credentials and Decentralized Identifiers, to granting authorization using SMPC for verification and access to EHR.
3) Conducts detailed security and privacy analysis of the security protocol using STRIDE [15] and LINDDUN [16].

The structure of this paper is organized as follows. Section 2 looks at related work and Section 3 provides the background for the SSI and SMPC-TC components of the proposed solution. Section 4 elaborates and discusses the system overview and design. Section 5 describes and elaborates the design of the SMPC-TC platform and section 6 discusses the implementation and the experimental results. Section 7 describes the threat analysis and discusses the security and privacy analysis. Finally, section 8 sets out the directions for future works and concludes the paper.

2 RELATED WORK

Currently, there are several researches conducted on the sharing of EHR using blockchain and different

cryptographic schemes and access control mechanisms for secure sharing and access to EHR on a blockchain and cloud platforms. The papers [17-22] propose utilizing the blockchain platform as a storage system for the access control model, protocols for authentication and sharing of healthcare data, and access control for shared medical records in cloud repositories, [23-28] propose a secure medical record sharing system using an attribute-based encryption and (multi-)signature scheme, [29, 30] propose a blockchain-based secure and privacy-preserving EHR sharing protocol using searchable encryption and conditional proxy re-encryption cryptographic schemes, [31] also uses searchable encryption but partitions the patient's record into a hierarchical structure, each portion of which is encrypted with a corresponding key, thus enables the patient to selectively distribute subkeys for decryption of various portions of the record, [32-34] present a communication-optimized secure inference protocol that purely relies on the lightweight secret sharing techniques and adapt secure multi-party computation framework to realize encrypted computation, and [35] proposes MedChain, which combines blockchain, digest chain, and structured P2P network techniques to provide a session-based healthcare data-sharing scheme.

The majority of the above literature solutions assume the data owner (e.g., the patient) to be available to approve before sharing the EHR, and only a few have taken into consideration the case whereby the patient is unavailable to perform the approval, e.g., in cases when s/he is unconscious in an emergency or mentally unfit to perform any tasks. The use of an 'allowed list' is proposed in [36] for clinicians to access a patient's data under emergencies via prior one-time authentication from the patient. But as it is under an umbrella account of the HSP that links all clinicians (i.e. shared account), data security and privacy of the patient can be a major concern, especially for those not involved in the patient's medical consultations. Using the concepts of organizational structure roles, [37] proposes to define entity-to-entity relationships and access rights based on functional roles and duties. This structure is used for authorization management as well as access control. However, this way of access control is specific to a pre-defined organizational structure and may not be aligned with the intention of the patient, the rightful data owner. Using their eTRON enterprise security architecture, [38] proposes a distributed system for delegation management that enables a patient to securely delegate access rights to her health records to someone s/he trusts. eTRON functions much like the SSI framework which has an issuer to issue an authorization token and this token is used for access to EHR. The solution requires hardware specific eTRON card with a chip that stores the holder's identity. Unlike SSI, whose building blocks components and standards are defined by W3C, eTRON is more propriety which may have interoperability issues for wide deployment. Attribute Based Encryption (ABE) is proposed in [39] to allow for delegated secure access to patient records. It similarly requires an organizational structure of the entities or stakeholders of a medical organization and its patients to map

out the access control rights based on the entities' attributes.

Self-Sovereign Identity (SSI), a decentralised technology for digital identity management, is a promising concept for handling health data. It could represent a step forward in empowering users, granting them control over their data [40]. A systematic literature review is conducted in [41] to investigate state-of-the-art measures based on SSI and Blockchain technologies for dealing with EHR. It concludes SSI is still a novel subject and, even though adopting the principles of SSI could make patient-centric solutions more accurate, current healthcare research has neither adequately defined nor employed it in the health context.

The solution proposed by this paper combines SMPC scheme to delegate the authorization and adopts the SSI principles and framework to ensure the validity and verification of the identities, credentials and claims. To the best of our knowledge, there is currently no related work on this approach.

3 BACKGROUND

3.1 Self-sovereign Identity

Self-Sovereign Identity (SSI) is a concept to empower an individual with the autonomy and authority to control her/his own personal and distinguishable data. Its solution is to build a decentralized user-centric ecosystem that grants an entity (an individual, an organization, or a device) the right to control, authorize and consent to the disclosure and usage of its own digital identity or credentials in order to fulfill a transaction. It can be considered as a digital identity framework. Decentralized Identifier (DID) and Verifiable Claims /Credential (VC) are the key essential building blocks of the SSI framework. DID is a type of verifiable identifier for self-sovereign digital identity that is universally discoverable and interoperable across a range of systems that conforms to standards defined by The World Wide Web Consortium (W3C) [42] - analogous to a digital certificate issued by a certificate authority [43]. It is an URL (i.e., unique web addresses) associated with at least one pair of cryptographic keys: a public key and a private key. Together, the DID and public keys are published in the blockchain, and this "package" is called a DID document which is the source of authentication. A DID Document provides information on how to use the specific DID. For example, a DID Document can specify that a particular verification method (such as a cryptographic public key or pseudonymous biometric protocol) can be used for authentication. A DID alone is only useful for authentication, but it becomes particularly useful when used in combination with verifiable claims or credentials (VC), another W3C standard, that can be used to make any number of attestations about a DID subject [44]. These attestations include credentials, claims and certifications that grant the DID subject access rights or privileges. A verifiable credential contains the DID of its subject (e.g., an HSP) and the attestation (access

approval) which must be digitally signed by the entity making the claim using the private key associated with the DID of the issuer (e.g., the patient) which issues the claim. Verifiable credentials are thus methods for trusted authorities (parties) to provably issue a certified credential associated with a particular DID to grant consent. It also guarantees privacy by enabling methods such as minimum/selective disclosure.

3.2 Secure Multi-Party Computation (SMPC)

Secure multi-party computation (SMPC) protocol, such as oblivious transfer [45], homomorphic encryption (HE) [46] and the secret sharing scheme (SSS) [47], provides enhanced privacy, correctness and independence of inputs, and guarantees output delivery. It suits a distributed network like blockchain as it deals with security and trust issues in distributed environments. It is helpful in the scenarios whereby confidential data are to be shared across several organizations, across several sources and to run some kind of joint aggregation analysis or processing. Only specifically crafted shards of the data are exchanged and every shard reveals nothing about the original data and it alone cannot be used to restore to the original. However, the joint processing of shards is still possible to analyze the data. SSS is a form of multi-party computation, whereby a secret is divided into parts, giving each participant its own unique part. To reconstruct the original secret, a minimum number of parts, known as the threshold, is required. In this threshold cryptography scheme, this number (***t***) is less than the total number of parts (***n***), otherwise all participants are needed to reconstruct the original secret. The secret sharing scheme defined in [48] is as follows:

Let P be a set $\{P_1,\ldots,P_n\}$ of n entities, called participants, who take part in sending and receiving communications. An access structure on P is a collection A of subsets of P. A subset $A \in A$ is called an authorized set (of participants). Thus any set of participants that contains an authorized subset is authorized. In a monotone access structure, a minimal authorized subset is a subset $A \in A$ such that $A\backslash\{a\} \notin A$ for all $a \in A$, and a maximal unauthorized subset is a subset $A \notin A$ such that $A \cup \{a\} \in A$ for all $a \in P \backslash A$.

For $i = 1,\ldots,n$, let S_i denotes a set of elements, called shares corresponding to participant $P_i \in P$. A secret sharing scheme for the key set K on the set of participants P is a subset D of $K \times S_i \times \ldots \times S_n$ together with a probability distribution defined on D. If $(k, s_1,\ldots, s_n) \in$ D then we say that key k is shared among the participants $P_1,\ldots,P_n$ who hold shares $s_1,\ldots, s_n$ respectively. The probability distribution on D induces a probability distribution on K and each of S_i, $i = 1,\ldots,n$. The set D is a secret sharing scheme for K with respect to the access structure A on P if

$$H(K \mid S_{i1},\ldots,S_{it}) = 0 \text{ iff } P_{i1},\ldots, P_{it} \in A$$

$$> 0 \text{ iff } P_{i1},\ldots, P_{it} \notin A$$

for all subsets $\{P_{i1},\ldots, P_{it}\} \subseteq$ P and shares $s'_{i1},\ldots, s'_{it}$ where $s'_{ij} \in S_{ij}$ for $j = 1,\ldots,$ t there is a $k' \in K$ such that $k =$

k' for every $(k, s_1,\ldots, s_n) \in D$ with $s_{ij} = s'_{ij}$ for $j = 1,\ldots,t$ if and only iff $P_{i1},\ldots, P_{it} \in A$. We say that an authorized subset of participants $P_{i1},\ldots, P_{it}$ pool their shares $s'_{i1},\ldots, s'_{it}$ to get the key k'.

Secret sharing schemes defined on ***n*** participants, whose access structure consists of all sets of size of at least ***t*** are referred to as (***t, n***)-threshold schemes.

In [48], a solution to provide shared encryption (decryption) by applying the secret sharing techniques to the sharing of block cipher is proposed. It uses 2 techniques, cascade and XOR, for the composition of block ciphers. When an authorized group wishes to encrypt a message or decrypt a ciphertext they cooperate by taking part in a protocol that enables them to perform the distributed computation of the cipher. This is an approach this paper adapts.

4 System Overview and Design

4.1 Solution Overview

The focus of the proposed solution is on the delegation of authorization to HSP to access the patient's own EHR if the patient is unconscious or mentally unfit to approve for immediate medical care to proceed. Our proposed system involves multiple parties with roles as (in the context of the SSI concepts) issuer, holder and verifier. In addition, these parties are also data owners, data custodians and data requesters. Fig. 1 illustrates the high-level view of the process flows and the parties involved. The roles defined are:

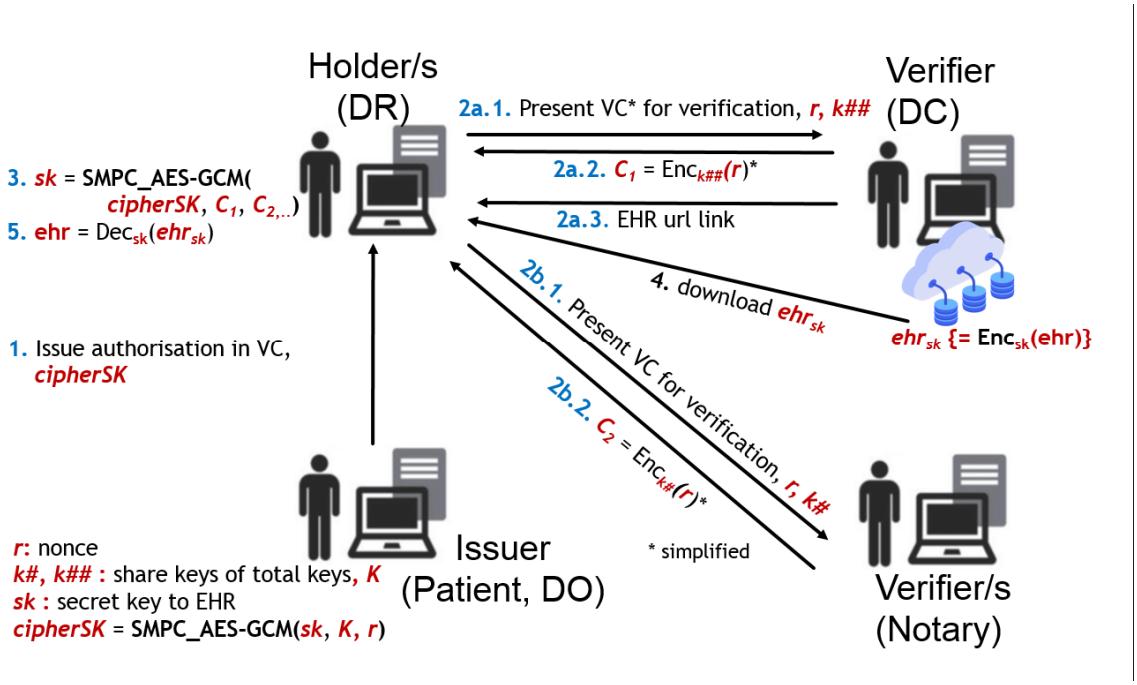


Fig. 1. A high-level view of the approach's flow and parties involved

- **Issuer**. The issuer, in our case the patient and as a data owner (DO), wishes to authorize beforehand the access

to her/his personal EHR to pre-delegated parties (data requesters - DRs) by issuing the verifiable credential (VC) to the DRs, The VC encapsulates the authorization details, the encrypted secret key to DO's EHR and other details.

- **Holder**. The holder, in our case the HSPs and as data requester (DR), stores the VC in its internal repository or digital wallet and uses the VC when s/he requires access to the issuer's EHR by presenting the VC as a claim that s/he is authorized. The holder is DO's pre-delegated parties to have access to the EHR.
- **Verifier**. The verifier, in our case the notary and cloud storage provider (CSP) as data custodian (DC) of the EHR, verifies the validity, authenticity and correctness of the VC presented by the holder (DR) before providing the necessary details for the access to the EHR.

Each of the participants in this ecosystem, whether it is an individual (e.g. patient) or an entity (e.g. organization like HSP), is issued a DID by an identity provider (IdP). The DID is also recorded on the blockchain (BC). HSP typically has a copy of the patient's EHR in their local system. Patient, as the data owner (DO) of the EHR, can request for her/his EHR to be accessible externally, e.g. via CSP. HSP encrypts the EHR and stores it in CSP, which becomes the data custodian (DC) of the encrypted EHR. HSP provides DO with the secret key, ehr-id and DC identity. DO will generate a set of unique keys according to the number of participating verifiers (parties, n) and the minimum threshold verifiers (threshold parties, t) needed to reveal the secret key. DO will encrypt the secret key with the set of unique keys. This set of unique keys is then split partially to the n parties whereby t parties will have all the set of unique keys required to decrypt and derive the secret key.

To ensure the validity of the authorization process, one or more Notaries are identified as the participating parties acting as verifiers/witnesses. The Notary can be a lawyer or a trusted independent party. This is analogous to the Power of Attorney (Lasting Power of Attorney) process whereby a lawyer and identified witnesses are involved [49] in the verifying of documents and authorizing consent. The set of unique keys is split and allocated among the *n* parties (verifiers) by encrypting the keys with their respective public keys, which are recorded in their DIDs. For transparency, accessibility and ease of key management, the ehr-id and the encrypted key sets are recorded on the blockchain (BC) using DO generated pseudo ID. DO can now issue a signed verifiable credential (VC) delegating who is authorized to access her/his EHR, with details of the verifiers, DC, ehr-id, DO's pseudo ID, expiry date, the required threshold verifiers (*t*), details of the set of keys allocation, the encrypted secret key, etc. By using a new pseudo ID every time on BC, DO's privacy can be protected from association or inference threats. The VCs are cryptographically signed by the DO and issued to authorized parties - the data requesters (DRs). The VC provides DR with the claim that it is authorized by DO for the access to the EHR identified by

ehr-id and only the VC contains the link between DO's DID and pseudo ID. When DR, holder of the VC, needs access to DO's EHR, s/he uses the VC to know the required verifiers (*t*) and disclose the essential details to the *t* verifiers to verify the authorization claim. The verifier can decide based on the validity and expiry of the VC details that have the DO's endorsement, and a check of the revocation lists. Once the verifier has validated the VC, it uses its allocated keys to perform the required multi-party computation that can eventually facilitate DR to decrypt the encrypted secret key stored in DO's initial VC. Since only DC knows the storage location of DO's EHR linked to the ehr-id, DR needs to also use the VC to request DC to provide the link to access the encrypted EHR. DC can impose a time period for the access – e.g. availability of the link. DR can download the encrypted EHR and decrypt it with the secret key to access the EHR content in clear.

In an SSI model, DO is the issuer, Notary is the verifier and DR is the holder of the VC. The VCs are presented via Verifiable Presentation (VP) and with VPs, the holders (for our case, DR) can freely choose which information (from the underlying VCs) to include in the VP and share with the verifier. This is the selective disclosure feature of the SSI solution and is further elaborated in the next section. Additional access rights and attributes can be defined in the VC to provide more fine-grained access control of the EHR content, e.g. using attribute-based encryption techniques.

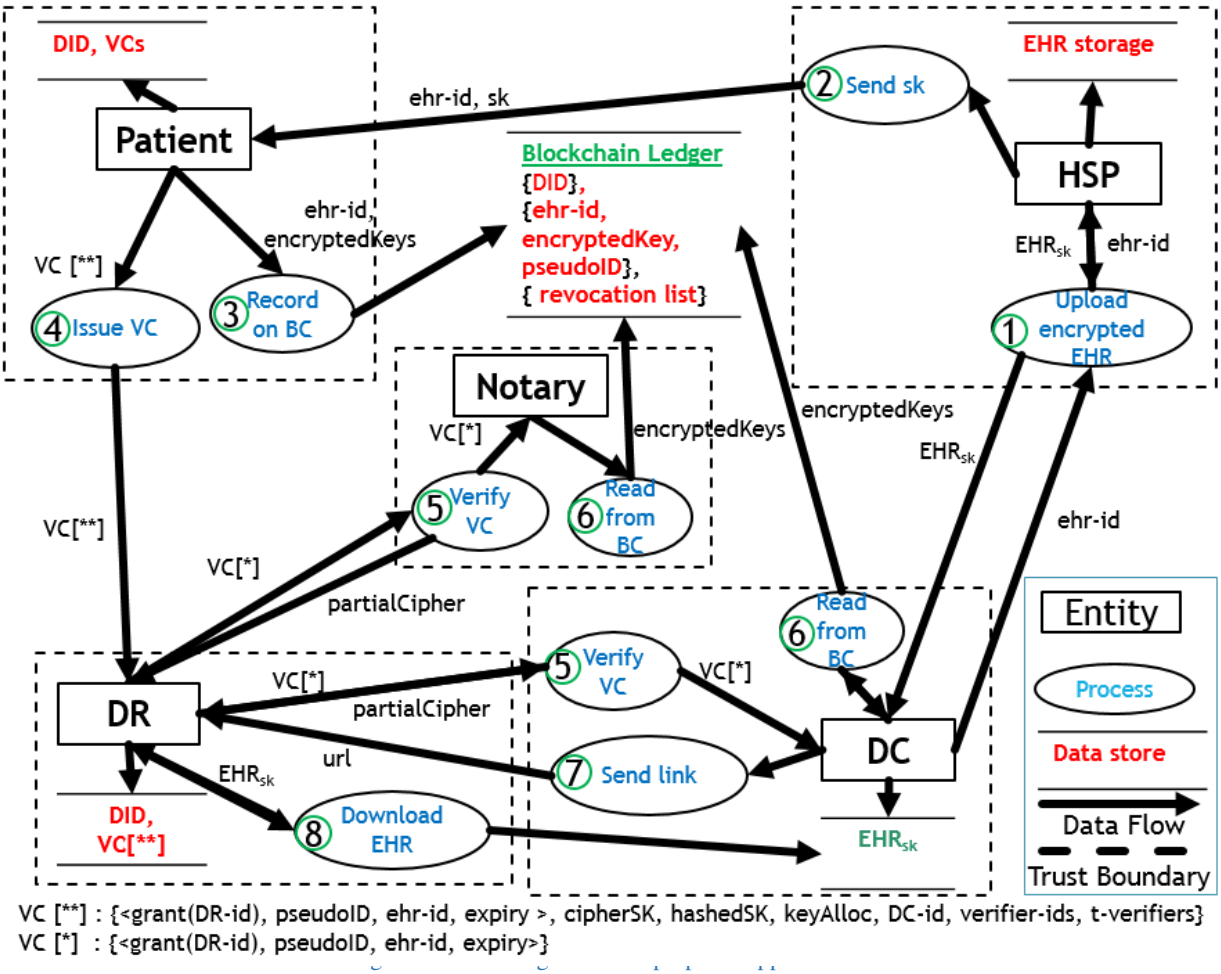
Note that DC, as the cloud storage provider (CSP) holding DO's encrypted EHR, need not be one of the choices as a verifier. It is selected to illustrate it can play a dual role since it has custody of the encrypted DO's personal EHR and thus the convenience in minimizing communication overhead from the DR aspect. This approach is not uncommon as was elaborated in reviewed literature proposing the proxy re-encryption (PRE) [50] approach, as an example, with the CSP as a semi-honest party in the process. The security and privacy analysis in the section 8 will also validate the viability of this choice. However, in consideration if DC is not the verifier party, DC may need further proof to authenticate the VC provided by DR and be convinced it is with HSP (the co-DO) consent/awareness. One way this can be achieved is through the use of a blind signature. Using blind signature, DC can validate HSP signature but without knowing to whom the EHR belongs.

4.2 Functional Flow

The functional flows are broken down into 4 phases, namely Setup, Secure storing of EHR, Delegation of authorizations to DRs and Verification and Secure Access to EHR. Fig. 2 shows the data flow diagram to illustrate the process flows of each of the phases which we elaborate below.

(a) Setup

Each of the participants, whether it is an individual (e.g. patient) or an entity (e.g. organization like HSP), needs



to register with the operator of this healthcare ecosystem (IdP) with their identities and credentials validated before a trusted Certificate Authority issues the participant with a digital certificate. The digital certificate has a cryptographic key pair (public key and private key) whereby the public key is encapsulated in a decentralized identifier (DID) used for identifying and authenticating the participant, aka its digital identity in this system. The DID is recorded on the blockchain (BC). The private key corresponding to the public key is safely and securely stored in the participant's digital wallet. HSP has its patients' EHR in its private storage and internal system.

(b) Secure storing of EHR (Process flow 1 and 2)

Alice, as a patient, has consulted a medical practitioner from an HSP and her EHR is recorded in HSP's private data store. The HSP is thus a co-data owner (co-DO) of the EHR. *Alice* requests the EHR to be made available to her.

(i) HSP retrieves the *Alice's* (DO) ***EHR*** from its private data store and encrypts it with a secret key, ***sk,*** before uploading it to a public cloud storage provider (DC). HSP is provided with ***ehr-id*** which is used to locate the encrypted EHR (***EHR$_{sk}$***) in DC's data store.

(ii) HSP encrypts ***sk*** and ***ehr-id*** with DO's public key (which is within DO's DID) and sends to DO - optionally

through a secure channel, e.g. Transport Layer Security (TLS) since it is already encrypted.

(iii) DO decrypts with its private key (from DO's DID in its digital wallet) and stores the ***sk*** and ***ehr-id*** in its digital wallet

(c) Delegation of authorization to DRs (Process flow 3 and 4)

DO wishes to pre-delegate parties with the authorization to access her EHR in the event she is unfit to do so.

(i) DO identifies the parties (DRs) to which it would like to grant the authorization to access its EHR, the total verifiers (***n***) and the threshold verifiers (***t***) to be involved.

(ii) DO generates a set of keys, k_j, (where j= 1,2,….x and $x = C_{t-1}^{n}$, the total number of keys depending on ***n*** and ***t***) and a nonce, ***r***, and encrypts ***sk*** with k_j and ***r*** using our AES-GCM crypto-platform to derive the encrypted secret key, ***cipherSK***. Thus,

$cipherSK$ = AES-GCM($Algo_{1,2}(r, k_j)$, sk)

We will elaborate the key generation, $Algo_{1,2}$,and the AES-GCM crypto-platform in a later section.

(iii) DO splits all the keys and allocates to each of the verifiers *y* keys, where $y = C_{t-1}^{n-1}$, and encrypts them using their public keys (retrieved from their DIDs) to derive, $encryptedKeys_i$, where i =1,2,….,n and is the verifier index.

(iv) DO generates a ***pseudoID*** for BC and records all $encryptedKeys_i$, its ***pseudoID*** and ***ehr-id*** on BC.

(v) DO generates for each authorized DR a verifiable credential (***VC*****) and inputs the authorization grant for each DR with its DID (***grant(DR)***), ***verifier-ids*** list, ***cipherSK, hashedSK, keyAlloc***, ***t-verifiers***, ***pseudoID***, ***ehr-id*** and ***VC*** expiry date, where ***hashedSK*** is the cryptographic hash of ***sk*** for integrity check, ***t-verifiers*** is the minimum verifiers to seek and ***keyAlloc*** is the key allocations of the verifiers. Thus,

VC** = {<***grant(DR-id), pseudoID, ehr-id, expiry >, cipherSK, hashedSK, keyAlloc, DC-id, verifier-ids, t-verifiers***}

(vi) DO digitally signs each ***VC***** using its private key, and encrypts the data using each DR's public key that is encapsulated in DR's DID.

(vii) DO issues the ***VC****s to the DRs, optionally through a secure channel since ***VC***** is encrypted.

(viii) Each DR decrypts its ***VC***** with its private key and stores the ***VC***** in its repository.

(d) Verification and Secure Access to EHR (Process flow 5 to 8)

If DO is unconscious and unable to authorize the access to her EHR:

(i) DR needs access to DO's EHR.

(ii) DR retrieves ***VC***** from its repository and reads DO's ***grant(DR), pseudoID***, ***cipherSK***, ***verifier-ids, t-***

verifiers, keyAlloc and ehr-id. With ***t-verifiers***, ***verifier-ids*** and ***keyAlloc***, DR knows the minimum verifiers (*t*) to seek verification, their DIDs, and what keys to use (***k#***) for encryption.

(iii) DR extracts nonce, ***r***, from ***cipherSK***.

(iv) DR selects and identifies the *t* verifiers in the ***verifier-ids*** list that are not blacklisted or on the revocation list and initiates a request to each of the *t* verifiers (V_h where h =1,2,….,*t*) to present the ***VC**** as a verification presentation (VP_h), disclosing only the relevant details: DO's signed ***grant(DR)***, ***pseudoID***, ***ehr-id*** and ***VC-expiry***, as well as additional details ***r*** and $k\#_h$. Thus,

VC^*= {<***grant(DR-id), pseudoID, ehr-id, expiry***>, ***r***, $k\#_h$}.

VC* is encrypted with the verifier's public key andsigned by DR for authenticity.

(v) Each verifier decrypts VP_h and validates ***VC**** contents, specifically, DO's authorization - ***grant(DR)***. [Verifier can also optionally verify DO's state and availability offline. If DO is available, the verifier can seek DO's approval.]

(vi) In addition, the verifier also validates the authenticity and expiry of ***VC**** as well as a check against the revocation list available on BC. A notification is sent to DO and this activity is recorded on BC.

(vii) If granted, the verifier reads from BC its encrypted keys ($encryptedKeys_i$) based on DO's ***pseudoID*** and ***ehr-id***. The verifier decrypts $encryptedKeys_i$ with its private key to derive its keys and encrypts ***r*** with the required keys according to $k\#_h$ to derive $partialCipher_h$. Thus,

$partialCipher_h$ = $Algo_{1,2}$ (***r***, $k\#_h$), where $Algo_{1,2}$ is either Algorithm 1 or 2 which is elaborated in the next section. $partialCipher_h$ is encrypted with DR's public key and returned to DR.

(viii) After gathering all $partialCipher_h$ from V_h, DR can present DO's authorization ***grant(DR)*** in the ***VC***** to DC to request the link to download DO's encrypted EHR, EHR_{sk}.

(ix) If DC is not one of the verifiers, it will validate the authenticity of the presented ***VC*** against DO's signature and provide the link to DR once validated.

(x) DR accesses the link to download the encrypted EHR, EHR_{sk}. DR decrypts the encrypted $partialCipher_h$ using its private key and performs the same crypto operation as DO - using the SMPC crypto-platform operation on all received $partialCipher_h$ with ***cipherSK*** to derive ***sk.*** That is,

sk = AES-GCM($Algo_3$ (***partialCipher***$_h$), ***cipherSK***),

where h =1,2,….,*t and* $Algo_3$ is Algorithm 3 which is elaborated in the next section.

(xi) Finally, DR can use ***sk*** to decrypt EHR_{sk} and retrieve *Alice's* ***EHR*** records.

Though not mentioned in the above functional flow, all critical activities in the flow that requires accountability,

e.g. authenticity of identity, proof, and validation of authorization of the sharing, etc, are logged on the blockchain and verifiable to ensure non-repudiation.

5 MULTI-PARTY COMPUTATION DESIGN AND IMPLEMENTATION

In a distributed system or operation, resilience and scalability are the primary considerations for reliability and performance. While maintaining a distributed system effectively requires multiple entities to collaborate in operation, having a provision for unavailability (or failure) of a limited number of entities in practice is crucial in such a case. Thus, standard distributed systems and operations adopt the principle of scalability with redundancy to address this issue. For distributed systems offering cryptographic services, data privacy and confidentiality are also a concern that needs consideration as the failure of an entity may cause data leakage in the system. In our approach, we propose the use of Secured Multi-party Computation (SMPC) to prevent data leakage and ensure data privacy and confidentiality, as well as Threshold Cryptography (TC) to ensure fault tolerance and provide resilience. The goal is also to distribute the computation of basic cryptographic primitives across several parties/nodes in order to relax trust assumptions on individual parties/nodes. In considering performance, although public (asymmetric) key encryption schemes are easier to use and administrate, they are slower than symmetric key encryption schemes, thus we adopt the latter. In our approach, sets of keys are allocated to each party (verifier entity) and a minimum set (threshold) of these parties compute using their allocated keys to generate a block cipher and share it with the requester (for our case, DR). With the gathered block ciphers, the requester performs the decryption of the message (secret key) using AES-GCM (Advanced Encryption Standard-Galois/Counter Mode) [51].

In this section, we elaborate in detail the SMPC (TC) module design and implementation, the different security approaches taken, and discuss the experimental results and performance. There are three main parts to consider in the design and development of this module - Key Management (allocation and distribution), Sharing Block Cipher Encryption and Encryption, and Decryption of Message using AES-GCM. We elaborate them in the next sections.

5.1 Key Management

(t, n) threshold scheme allows for information distribution within n parties such that the relevant information is only available to subsets of minimum size t. Therefore, (t -1) or a smaller number of parties will not be able to recover the information, even if they collude [52]. The operation of our approach is single block-wise encryption or decryption using AES (Advanced Encryption Standard) symmetrical block cipher algorithm, the information to be distributed amongst the parties is the set of keys used in the encryption and decryption functions. We next

elaborate the key allocation and distribution mechanism in line with (*t, n*) threshold cryptography to provide the security guarantees in the system.

Key Allocation

Key allocation to each user in an (*t*, *n*) scheme is based on the identification of maximal unauthorized subsets and each subset should contain (*t*-1) parties. We have C_{t-1}^{n} different subsets of this kind. Thus, one needs to generate C_{t-1}^{n} keys and distribute C_{t-1}^{n-1} keys to each party in the (*t, n*) scheme. In the *t*-party computation designed in the next section, the parties will use their key shares to encrypt or decrypt data without ever having to produce or recreate a whole key during any individual round of operation. [Assuming each party is a verifier, V_i]

Example. Medium scale use-case: $t = 3$, $n = 5$ with verifier (V_1, V_2, V_3, V_4, V_5).

Maximal unauthorized subsets are as follows:

(V_1, V_2), (V_1, V_3), (V_1, V_4), (V_1, V_5), (V_2, V_3), (V_2, V_4), (V_2, V_5), (V_3, V_4), (V_3, V_5), (V_4, V_5).

We need to choose a total of C_{3-1}^{5}= 10 random keys (k_1,k_2,k_3,k_4, k_5,k_6,k_7,k_8,k_9,k_{10}) and associate each subset with one key. If a verifier is not contained in a subset, then that verifier is assigned with the secret keys corresponding to the subset. In this case, the resultant key allocation for the 5 verifiers is as follows in TABLE 1.

TABLE 1

KEY ALLOCATION EXAMPLE FOR N=5, T=3

	(V_1,V_2) k_1	(V_1, V_3) k_2	(V_1,V_4) k_3	(V_1,V_5) k_4	(V_2,V_3) k_5	(V_2,V_4) k_6	(V_2,V_5) k_7	(V_3,V_4) k_8	(V_3,V_5) k_9	(V_4,V_5) k_{10}
V_1					√	√	√	√	√	√
V_2		√	√	√				√	√	√
V_3	√		√	√		√	√			√
V_4	√	√		√	√		√		√	
V_5	√	√	√		√	√		√		

As per the allocation based on maximal unauthorized subsets and using C_{t-1}^{n-1}, each verifier will receive C_2^4 =6 keys as the follows:

$V_1 = (k_5, k_6, k_7, k_8, k_9, k_{10})$, $V_2 = (k_2, k_3, k_4, k_8, k_9, k_{10})$,

$V_3 = (k_1, k_3, k_4, k_6, k_7, k_{10})$, $V_4 = (k_1, k_2, k_4, k_5, k_7, k_9)$ and $V_5 = (k_1, k_2, k_3, k_5, k_6, k_8)$.

Key Distribution

The key generation and allocation are performed by DO and provided to DR in the VC. The key distribution is also performed by DO since DO has the keys and needs to identify the verifiers to delegate the decentralized verification of authorization and provide DR with the partial cipher. The key distribution is performed by encrypting the partial set of keys that are allocated to the verifier with the verifier's public keys and stored on BC as shown in Fig. 3.

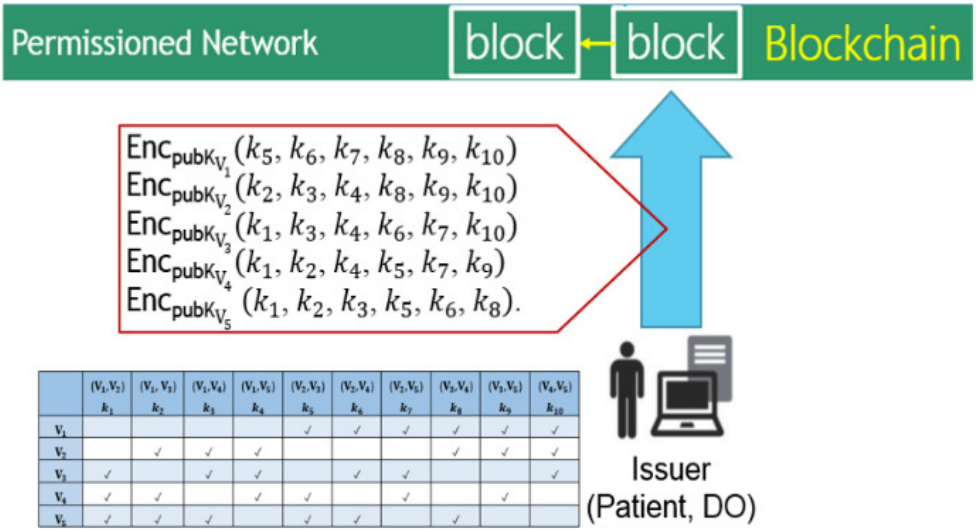


Fig. 3. Key distribution example to 5 verifiers recorded on the blockchain

5.2 Sharing Block Cipher Encryption

In a t out of n threshold SMPC setting, where $1 \leq t \leq n$, any t out of n users can work together to encrypt or decrypt a message m. In a case of a collusion set less than t, it is impossible to decrypt the ciphertext that has originally been encrypted by t users. As part of this research, we adopt the model for shared encryption/decryption of a symmetric key block cipher proposed by Martin et al. [48]. Sharing a block cipher refers to the process in which the encryption or the decryption of a message sent using that block cipher is distributed amongst a group of entities. For example, in shared encryption, a group of senders co-operatively compute the ciphertext, which they then send to a single /receiver, who decrypts it or uses it for other cryptographic functions. An access structure can also be specified to associate the distributed end of the group of senders as authorized subsets. The main benefits of sharing a block cipher computation are that we can distribute the trust required to perform the computation to a group, to co-operate in the shared encryption (and decryption), and the increased availability of distributed nodes to operate in the presence of faults in other nodes, thus avoid the "single points of failure" threat.

In this section, we present two mechanisms using this sharing of block cipher computation– Cascade and XOR, and implement them to generate experimental results. We elaborate each of the mechanisms and argue that the t-party computations are as secure as standard encryption/decryption with the underlying block cipher. We also compare the two mechanisms (Cascade and XOR) to identify the better of the two in terms of performance and computation time with similar security margins.

Cascade Mechanism

We adapt the idea of sequence sharing proposed in [53] that is based on the composition of block ciphers using the technique of cascading.

In cascade encryption, the requester (DR) performs the following steps:

1. Identification of the first k_i from the V_1 till V_t, where $i \in \{1,2,\dots l\}$, $l = C_{t-1}^{n}$, the total unique keys.
   a. Choose V_j with the longest consecutive sequence of keys, starting from current k_i, where $j \in \{1,2,\dots t\}$
   b. Otherwise, choose the existing V_j with k_i
2. Encryption (E_{k_i}) of random nonce (r) or ciphertext of previous encryption by using the key k_i.
   a. If this verifier has consecutive keys k_{i+1}, k_{i+2},…, the nonce (or ciphertext) is encrypted using these keys
   b. Repeat step 1 till the last key sequence

Thus the block cipher is derived as:

$$E_k(r) = E_{k_l}(\dots E_{k_5}(E_{k_4}(E_{k_3}(E_{k_2}(E_{k_1}(r)))))) \quad (1)$$

As an example, using our initial key allocation, if V_2, V_3 and V_5 are the selected verifiers, V_5 is selected to start with keys k_1, $k_2$and k_3 since it has the longest consecutive sequence, followed by V_2 and so forth. Fig. 4 shows the verifiers and key sequences for our cascade example.

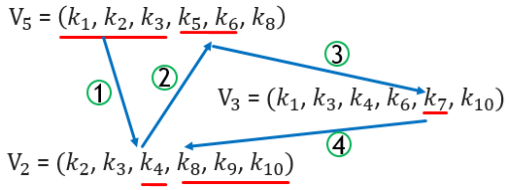


Fig. 4. Example of cascade key contribution by verifiers

The sharing of the encryption $E_k(r)$ provides public access to the result of the first application of the cipher E (under key k_1), and an attack on E provides an attack on the shared block cipher $E_k(r)$. So, the shared block cipher $E_k(r)$ is no more secure than E. In this case, the security of the sharing of block cipher $E_k(r)$ is equivalent to that of E.

XOR Mechanism

In XOR encryption, each verifier encrypts a random nonce r provided by the requester by using the keys specified by the requester and sends the result back to the requester. Finally, the requester combines (XOR) all the results together.

Thus the block cipher is derived as:

$$E_k(r) = E_{k_l}(r) \dots \oplus E_{k_5}(r) \oplus E_{k_4}(r) \oplus E_{k_3}(r) \oplus E_{k_2}(r) \oplus E_{k_1}(r) \quad (2)$$

As the XOR encryption has no restriction on the key sequences, the requester has the flexibility to take different approaches to decide the key contributions from the t verifiers. For example, equal contributions or the most trusted to use all its allocated keys. As total key contribution directly affects the total encryption time taken for the verifier, this needs to be taken into consideration when deciding the approach to take.

Since the cipher $E_k(r)$ is the l-fold composition of commuting ciphers, the result of Maurer and Massey [54]

shows that when the component keys are independent, $E_k(r)$ is as secure as $E_{k_1}(r)$. Thus, the sharing of block cipher $E_k(r)$ is also as secure as block cipher E.

5.3 Encryption and Decryption of Message using AES-GCM

After each selected party (V_i) computes the block cipher, it sends the block cipher back to the requester (DR). The requester is required to combine the block cipher with the encrypted message to derive the clear message. We have selected to use AES-GCM (Galois/Counter Mode) - a block cipher mode of operation that uses universal hashing over a binary Galois field to provide authenticated encryption.

In security and implementation aspects, AES-GCM has the following needed features as compared to other modes like ECB (Electronic Codebook) and CBC (Cipher Block Chaining) [51]:

a. It can efficiently provide message authentication through authenticated encryption with associated data (AEAD) at high speeds in hardware and perform well in software. The produced authentication tag can be used to verify the integrity of the message.
b. Its design is based on the security of the block cipher, which aligns with our approach of using shared block cipher.
c. It requires pipelined and parallelized implementations, and thus has minimal computational latency to be useful at high data rates. This can mitigate the resources required when implementing on mobile devices with limited hardware resources.
d. It accepts initialization vectors (IV) of arbitrary length which removes the restriction for applications to provide distinct lengths. In GCM, a nonce of any size can be used as the IV and can be prepended on the ciphertext as additional authentication data (AAD).

To integrate our multi-party (verifiers) block cipher sharing with the AES-GCM model, we implement a multi-party version of the AES-GCM as a full-fledged cryptographic platform, i.e., instead of the usual local single key stand-alone function call, we develop it as a platform for multi-party to use different keys and provide the inputs to the adapted AES-GCM flow. The adapted AES-GCM is shown in Fig. 5. For AES-GCM, the authenticated decryption operation is similar to the encrypt operation, but with the order of the hash and encrypt steps reversed, in addition to producing the authentication tag. Thus Fig. 5 is applicable for both encryption and decryption with the plaintext and ciphertext interchanged. Since for our approach, the encryption is done solely by one party (for our case, DO), we first describe the decryption process using this platform followed by the encryption.

The requester extracts the nonce (IV) which is prepended to the ciphertext and provides (sends) to the verifiers which use it for generating the counters (Ctr_0, Ctr_1, …, Ctr_x, where x is the number of blocks) and encrypts the counters with AES block ciphers using their assigned keys (as elaborated in section 5.2). This is shown as the green box in the

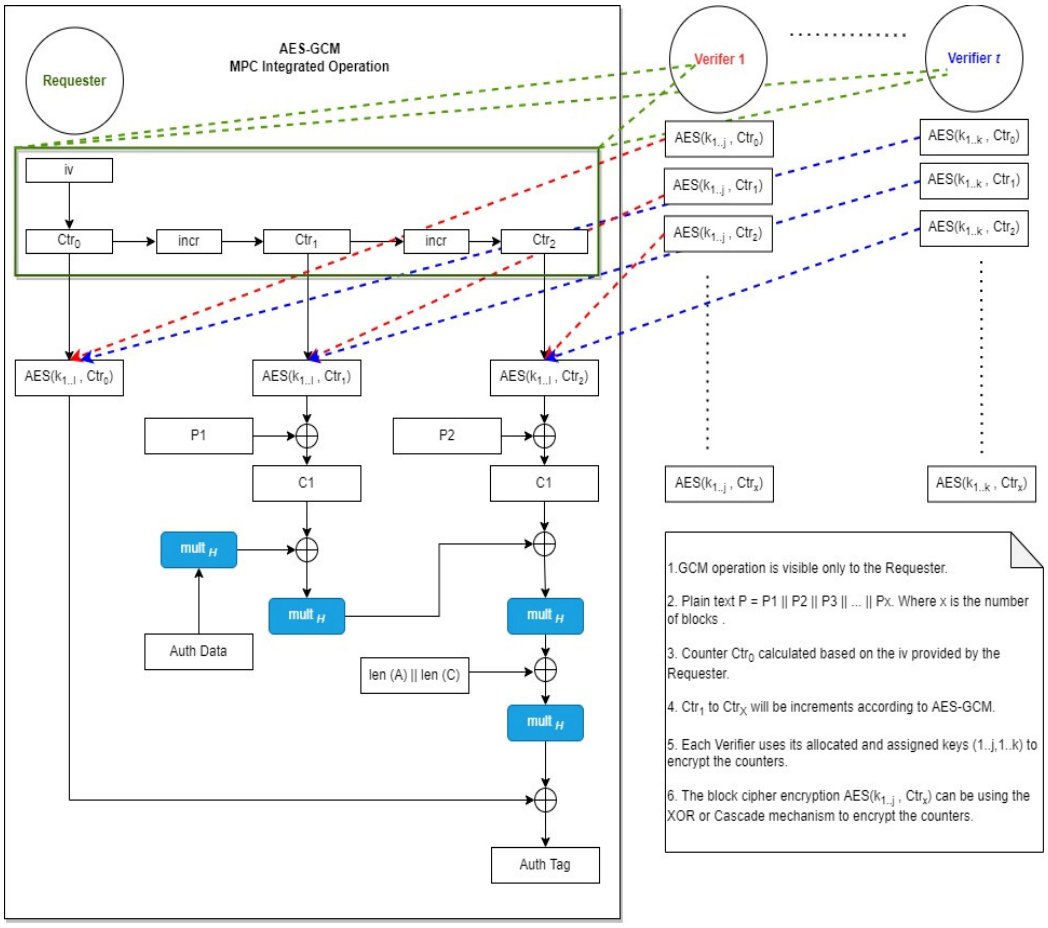


Fig. 5. AES-GCM SMPC integrated operation

figure. The results of these encryptions are returned to the requester which are then XOR-ed with the data blocks (P1, P2, …,P*x*) of the plaintext (ciphertext) to produce the ciphertext (plaintext) block. [Note that Ctr_0 is only used to generate the authentication tag and not with the data blocks]. Thereafter, the flow follows the usual AES-GCM steps to produce the complete ciphertext (plaintext) with the authentication tag (only for encryption).

For encryption of the plaintext (the secret key that encrypts the EHR), as DO does not need the verifiers to encrypt the secret key, the encryption is performed solely by DO using the full set of unique keys it has generated. So, instead of the multi-verifiers, DO generates a unique initialization vector (a nonce) and encrypts the counters with the AES block ciphers using the full set of unique keys which are then XOR-ed with the data blocks (P1, P2, …,P*x*) of the plaintext (secret key) to produce the ciphertext block. The algorithms for the XOR and Cascade mechanisms and the final derivation of the plaintext/ciphertext are elaborated in Algorithm 1, 2 and 3 respectively.

The purposes of Algorithm 1 and 2 are to encrypt the counters derived from the nonce or initiation vector (*iv*) and the plaintext/ciphertext length (*pl*) with the keys (*cryptoKeys*) to produce the block ciphers (*eCtr*) using the XOR and cascade mechanisms respectively. The plaintext/ciphertext length (*pl*) is used to derive the number of blocks and, for the XOR mechanism case (Algorithm 1), decide the XOR functions to use for full and partial blocks. As a ciphertext also encapsulates the authentication tag within and thus increases the ciphertext length,

IsDecrypt is used to reduce its length during decryption.

Algorithm 1 XOR mechanism for encrypting counters by a single peer

Input:
a. *cryptoKeys* : Map of keys <keyi> where i=0,….k, k is number of keys the peer has to use
b. *pl* : Plaintext/Ciphertext length
c. *iv* : nonce or Initiation Vector (IV)
d. *isDecrypt* : decryption flag

Output: *eCtr* : Map of encrypted counters <eCtr>

```
mask, out ← byte [GCMBlockSize]
counter ← deriveCounter(iv)              ► derive counter from nonce
blk← 0
if isDecrypt then pl ← pl - GCMTagSize   ► reduce by auth tag size
pl ← pl + GCMBlockSize                   ► offset for counter 0
while pl > GCMBlockSize do               ► for each full block
    for each key in cryptoKeys do        ► using each assigned keys
        blockCipher ← AES.NewCipher(key)
        mask ←blockCipher.Encrypt( counter)
        out ←XorWord(out, mask )         ► XOR the same counter
    eCtr[blk] ← out
    blk← blk + 1
    pl ← pl  - GCMBlockSize              ► decrement by block size
    counter ← incrCounter(counter)       ► increment counter
if pl > 0 then
        ► for less than full block
    for each key in cryptoKeys do
        blockCipher ← AES.NewCipher(key)
        mask ←blockCipher.Encrypt(counter)
        out ←XorBytes(out, mask )
    eCtr[blk] ← out
return eCtr
```

Algorithm 2: Cascade mechanism for encrypting counters by a single peer

Input:
a. *cryptoKeys* : Map of keys <key_i> where i=0,….k, k is number of keys peer has to use,
b. *pl* : Plaintext/Ciphertext length
c. *iv* : nonce or Initiation Vector (IV)
d. *eCtr :* Map of encrypted counters <*eCtr*>
e. *isDecrypt* : decryption flag

Output: *eCtr* : Map of encrypted counters ***<eCtr>***

```
if isDecrypt then  pl ← pl - GCMTagSize                ► reduce by auth tag size
if len(eCtr) == 0 then
        ► counters not created yet
    counter ← deriveCounter(iv)                        ► derive counter from nonce
    blk← 0
    pl ← pl + GCMBlockSize                             ► offset for counter 0
    while pl > GCMBlockSize  do                        ► for each full block
        eCtr[blk] ← counter
        blk← blk + 1
        pl ← pl  - GCMBlockSize                        ► decrement by block size
        counter ← incrCounter(counter)                 ► increment counter
    if pl > 0 then  eCtr[blk] ← counter                ► created counter-blk pairs
    for blk = 0 to len(eCtr)  do
        for each key in cryptoKeys do
            blockCipher ← AES.NewCipher(key)
            eCtr[blk] ←blockCipher.Encrypt(eCtr[blk])
return eCtr
```

For the Cascade mechanism (Algorithm 2), since the encryption of the counters has to be performed in key sequence from peer-to-peer and only the first peer holding the first key needs to create the counter, a check is first done to verify if the counters are created earlier in prior peer encryption. Thus, Algorithm 2 has an additional block cipher input, eCtr.

For Algorithm 3, the purpose is to XOR merge the plaintext/ciphertext data blocks with the counters block ciphers derived through either Algorithm 1 (XOR) or Algorithm 2 (Cascade).

Algorithm 3: XOR operation on the counter block ciphers with plaintext/ciphertext blocks

Input:
a. *eCtr :* Map of encrypted counters <$eCtr_i$> where i=0,….x, x is number of plaintext/ciphertext blocks,
b. *in :* Plaintext/Ciphertext, in bytes.

Output: *out:* Ciphertext/Plaintext

```
for i = 1 to len(eCtr) do          ► counter 0 is for authentication tag
  while len(in) > GCMBlockSize do
      out ←XorWord(in, Ctr_i )
      out ← out[GCMBlockSize : ]          ► slice off first GCMBlocksize of out
      in ← in[GCMBlockSize : ]            ► slice off first GCMBlocksize of out
  if len(in) > 0 then out ←XorBytes(in, Ctr_i )
return out
```

6 Implementation and Experimental Results

6.1 Implementation and Experiments

We implemented the SMPC AES-GCM platform using Golang version 1.17. Golang is chosen for its concurrency features as well as performance, as compared to a popular language like Java, since Golang functions much like C language. For a distributed system implementation, concurrency and speed are critical features to have. Our codebase has ~6500 lines of code (LOC). We made use of the libp2p library to build the peer-to-peer network and the communication protocol. To implement our SMPC AES-GCM cryptography platform, we adopted the crypto library provided by Golang, specifically, crypto/aes and crypto/cipher modules, and extended the internal codes of aes-gcm package with our intended algorithm implementations as described in the previous section. The network and crypto modules are integrated into a modular design that is extensible to 'plug-in' different key distribution and contribution methods. In this section, to relate better to the peer-to-peer network terms, we will use 'peer' instead of 'verifier'.

For our experiments, we compared the performance of the XOR and Cascade mechanisms using different total (n) and threshold peer nodes (t). The focus is on how the two mechanisms compare with (fixed t, varying n) and (fixed n, varying t), knowing the variation of t affects communication overhead from communicating (rounds) with more peer nodes and varying both n and t also affects total unique keys (=C_{t-1}^{n}) to be used and thus computation overhead (encryption and XOR). For the communication size, only the nonce or IV is sent to, and

the encrypted counters, depending on the plaintext/ciphertext size, are returned from the t peers. It is the same for the two mechanisms. A nonce length of 12 bytes (96 bits) is used, considering a 96-bit IV value can be processed more efficiently [51]. As for the plaintext (p) that is input to the SMPC AES-GCM cryptography platform, we have fixed it as 20 bytes, considering it is just a secret key and we assume a key size of 160 bits (20 bytes) is used. We installed and set up virtual machine (VM) instances on the cloud platform with the following specifications and parameters:

Test scripts were written to execute the encryption and decryption of the plaintext and ciphertext respectively for 1000 rounds and average values were calculated. We added code to collect the time taken for the function calls and remote procedure calls (RPC) over the network in order to measure the local CPU computation time and communication time respectively. The experiment results also verified the correctness of our SMPC AES-GCM cryptography platform implementation through the matching of the plaintext with the decrypted ciphertext after the sharing of block cipher encryption through t multiple peers.

Hardware & Software Resource Specification:

Ten virtual machine instances are created as nodes, each with an Intel Xeon @ 2.2GHz (2 Cores allocated) CPU with 4 Gigabytes memory and 16 Gigabytes internal storage with Ubuntu 16.04 operating system and Golang version 1.17 installed.

System parameters:

The system is configured to use a 96 bits nonce, 128 bits key size, 128 bits GCM block size and a plaintext length of 160 bits. The test scripts are executed for a total of 1000 rounds.

6.2 Experimental Results and Discussions

Fig. 6 and 7 show the time comparison of the two mechanisms with fixed t as 2 and varying n from 3 to 10, and fixed n as 10 and varying t from 2 to 9 respectively. With a fixed t(=2), the Cascade mechanism performs better than XOR as shown in Fig. 6. The required communication rounds are the same (to 2 peers) for the two

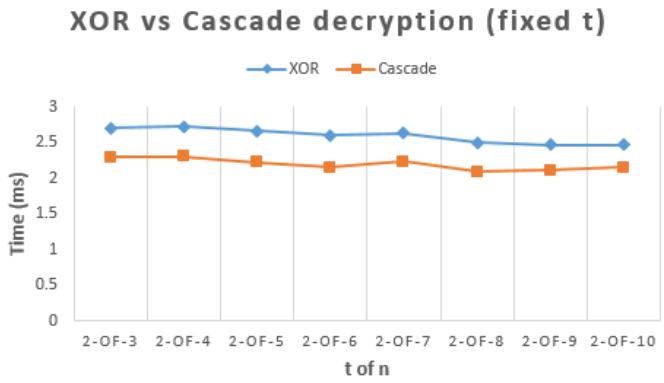


Fig. 6. XOR vs Cascade decryption comparison: fixed t, varying n

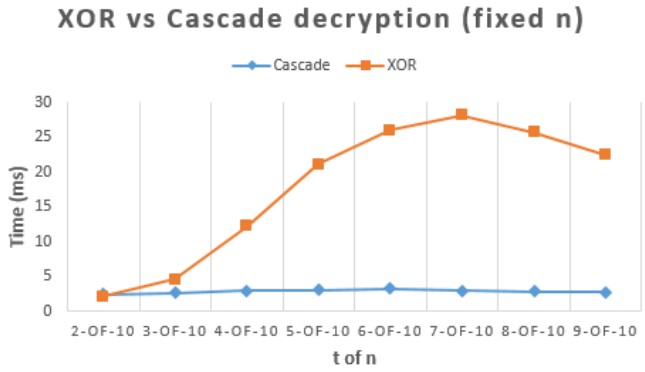


Fig. 7. XOR vs Cascade decryption comparison: fixed n, varying t

mechanisms, but the Cascade mechanism requires sequence sharing of the block cipher while the XOR mechanism can be parallelized and executed concurrently by the peers. However, XOR required additional XOR operation (refer to Eqn. 2 and Algorithm 1) after encrypting the counters with the block ciphers. The computation for Cascade is $C_{t-1}^{n} * E$ while XOR is $(C_{t-1}^{n} * E) + (C_{t-1}^{n} * X\,O\,\mathrm{R})$, E being the encryption of the counter. The performance result shows that the computation cost incurred by the XOR mechanism (i.e. the time taken for the additional XOR operations) is greater than the communication costs incurred by the Cascade mechanism. When the variation is reversed with fixed n as 10 and t varying from 2 to 9, the chart in Fig. 7 shows the significant increase in the communication cost for the Cascade mechanism as t increases, while the XOR mechanism stays approximately flat as it can be parallelized. The hump shape with a peak at 7-OF-10 (t=7, n=10) shows the compilation of the communication and computation times since, besides an increase in communication rounds, it also requires a larger number of keys ($C_{t-1}^{n} = C_{6}^{10} = 210$) for encryption. Though (t=6 and 8, n=10) requires the largest number of keys $C_{5}^{10} = 252$ keys (as maximum keys is when t -1≈n /2 since $C_{x}^{n} = C_{n-x}^{n}$), its total time taken is not at the peak. To understand this behavior, we break down the computation (CPU) time and communication (Network) time and present them in Fig. 8 and 9.

Fig. 8 shows the chart to analyse computation and communication time by comparing the two times and Fig.

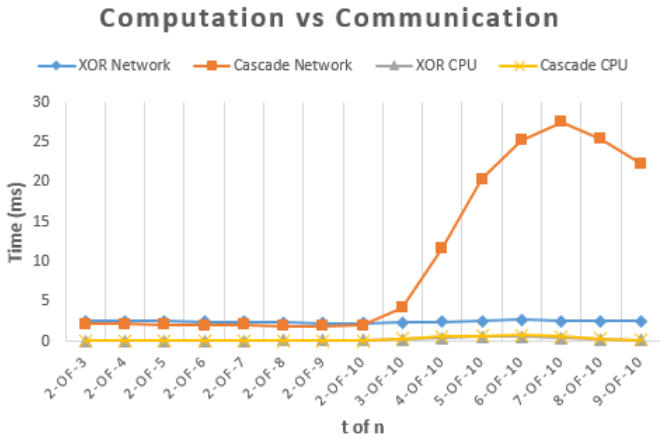


Fig. 8. Computation and communication times

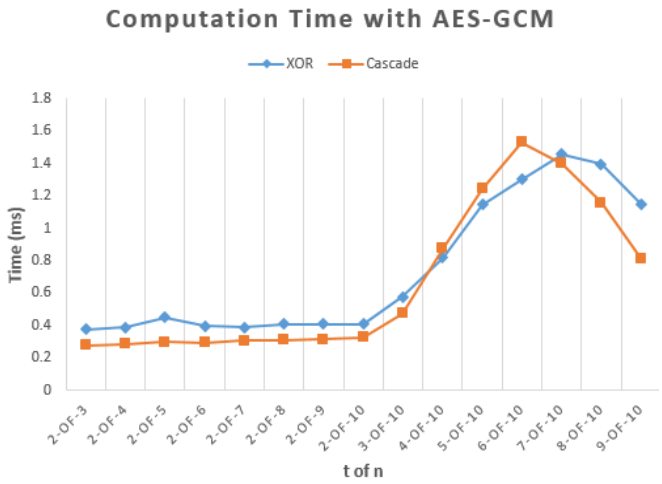


Fig. 9. Computation times for initiation and final AES-GCM

9 shows the chart focusing on the computation time. We omit the XOR operation time in the XOR mechanism so as to verify just the encryption computation time for the two mechanisms using the same number of keys. The communication time for Cascade and XOR are verified respectively as one took longer due to dependency on keys sequencing requirements and the other is almost flat as it can be executed in parallel concurrently. The communication time peak for Cascade was at 7-OF-10 (t=7, n=10), and comparing the computation and communication times differences between 7-OF-10 (with 210 keys same as 5-OF-10) and 6-OF-10 (with 252 keys), the communication overhead is much significant for the former and thus the overall time is taken. Going by the keys that are allocated to each peer for the cases of 5-OF-10, 6-OF-10 and 7-OF-10, each peer has 126, 126 and 84 keys respectively. So for 7-OF-10 case, this can be due to each peer having lesser keys (84) but needing a total of 210 keys to complete the process, therefore multiple same peers (within t) are needed to contribute to the sequence multiple times (as illustrated in Fig. 4) and, thus, the communication time increases. In terms of the computation time in the initiation to decide on the key contributions of the t peers and the final AES-GCM process to consolidate all peers' encrypted counters and produce the final plaintext, as shown in Fig. 9, for fixed t (=2), the XOR mechanism has the overhead of the additional XOR operations and with a maximum of 10 keys (C_{2-1}^{10}). But with varying t, the number of keys increases, and the initiation functions to derive the peer-key sequence becomes more significant for the Cascade mechanism, as compared to the XOR mechanism to use the keys in any sequence. All in all, though the readings gathered can also have minor margins of error and other network overheads (e.g. intermittent spikes), the outcomes are mostly within our expectations.

To sum up the overall results, for lower threshold (t) lesser than or three peers (verifiers), the Cascade mechanism can perform better than the XOR mechanism since it does not require the additional XOR operation computation overhead to merge the results after encrypting by using the assigned keys. But as the threshold

increases, the advantage of the XOR mechanism's parallelism implementation out-weights its XOR operation overheads in terms of computation time while the key sequencing restriction of the Cascade mechanism's communication overheads increases as more peer communications are needed and have to perform in sequence. On the initiation side which needs to derive the key contributions of the t peers, further optimization can be explored to improve the efficiency and performance of the algorithm for finding the peer-key sequencing for the Cascade mechanism. XOR mechanism has a better flexibility in this area to try different methods, e.g. share the keys equally, select the most trusted peers to use most of its allocated keys, etc.

In terms of communication cost, the complexity for the XOR mechanism is $O(n)$, where n represents the t - threshold of the selected peers since it just needs to communicate with the t peers to perform the encryption. For the Cascade mechanism, it has a minimum of t peers and depends on a specific scenario - the selected t peers and the keys they hold as well as the total unique keys required – as the same peer may be called more than once to contribute its key/s. This is inevitable for the Cascade approach due to its sequencing requirement with an example case of t=2 with one peer having odd keys and the other, having even keys. Our current implementation of selecting the peer with the longest consecutive keys may not be the most optimal method to minimise the rounds and a better method can be explored in future work.

In short, the XOR mechanism is recommended if more than three verifiers (as a threshold) are required to participate in the verification and cryptography operation. It incurs less communication cost with a similar security guarantee, and it can be done in parallel. Though it requires some additional XOR operations, which can be significant for lower threshold $t \leq 3$ peers (verifiers), and ciphertext storage (before the XOR operation), this contributes only little computation and storage cost when t increases to cater for more verifiers option. Thus, the XOR mechanism approach is selected and henceforth, our analysis of the overall system is with this mechanism in place.

7 THREAT ANALYSIS OF PROTOCOL AND DISCUSSION

Threat modeling is performed with a security analysis and a privacy analysis to illustrate the threats exhibited by the system functions. The data flow diagram in Fig. 2, which illustrates the data flow as was elaborated in the functional flow in the earlier section, is analysed for security and privacy threats. A discussion of the threats and how our proposed protocol addresses them follows.

7.1 Security Analysis

We use STRIDE [15] for the security analysis. We discuss the security threats and evaluate how it is addressed in

our solution.

- **Spoofing**:

(i) The identities of the participants are recorded on BC in the form of DIDs which stores the public keys which can verify the participant's identity and signature.

(ii) The corresponding private key is safely stored in the participant's digital wallet or repository which can be retrieved for cryptographic operations and generating a digital signature.

Threat: A likely spoofing of identity threat can be during validation and verification of the DID before recording onto BC which relates to the implementation of consensus protocol. However, as we are using a permissioned blockchain network with each participant's identity registered and linked to its DID and digital certificate, this threat can only happen if the Certificate Authority or the authority that validates the participant's identity is compromised.

- **Tampering**:

(i) The VC is digitally signed by the issuer which ensures the integrity and authenticity of the signed content since it requires the issuer's private key to do so.

(ii) DO's EHR is encrypted with a secret key known only to him/her and the HSP (co-DO). One additional measure is for DO to digitally sign the encrypted secret key after the secret key is encrypted with the set of generated keys. Currently, the hashed of the secret key is available.

(iii) DC will not be able to tamper with the EHR since it is encrypted and it is non-beneficial to itself, as a business entity providing service, to provide a tampered link to DR for downloading the encrypted EHR.

- **Repudiation**:

(i) The interactions between the participants are direct interactive and the verification of identities is immediate through validating their digital certificate, thus the participants cannot deny the interactions and actions taken.

(ii) VC is digitally signed and issued by DO to DRs.

(iii) Only DR knows how to derive the secret key to decrypt the data downloaded from the link provided by DC. Furthermore, DC can track the timestamp and location of the download as part of the audit and accountability policy that most storage providers have in place.

- **Information Disclosure:**

(i) Only pseudoID and ehr-id are recorded on BC, which are unlinkable to the participant's true identity and location of the EHR. The secret key to decrypt the encrypted EHR is only known to the DO and HSP (co-DO) and is in encrypted form encapsulated in the VC provided to authorized DR. DR derives the secret key

independently and locally.

(ii) DR can selectively disclose the need-to-know information when requesting verification, thus protecting the confidentiality of other information.

(iii) DC cannot disclose any content of DO's encrypted EHR other than its storage location.

(iv) The setting of the expiry date of VC and a revocation list recorded on the BC can mitigate this risk by removing the DR's access.

(v) As the verifiers independently compute the partial cipher (***partialCipher***$_i$) when requested by DR, each of their 'share' is not revealed to each other.

Threat: Only threat is HSP (the co-data owner) and DR's revelation of the EHR content after retrieving clear content. This can be mitigated through law-abiding policies like NDA or code of ethics. From a reputation perspective, it is not beneficial for them to do so.

- **Denial Of Service:**

(i) Except for DC which holds the location of EHR, all contents are on BC which is decentralized and available. However, this threat should be already mitigated by DC as most cloud storage providers have redundancy and resilience built-in in their environments.

- **Elevation of Privileges:**

(i) VC is only issued to the authorized DRs with details also recorded on the VC. An unauthorized party cannot use the VC as its own or download the encrypted EHR since the identity of the authorized DR is digitally signed by DO and encapsulated within the VC. The verifiers will have rejected the claim from the unauthorized party.

(ii) DC will have mitigated this risk as part of its security posture as a storage/service provider.

(iii) Multi-parties are involved with the identities of the DC and verifiers (which are selected from the verifier list) recorded in the VC created and signed by DO. A possible threat is collusion among the parties which is analysed next.

We further analyse the collusion and key management threats and evaluate our approach against them.

- **Collusion:**

(i) Verifier with unauthorized DR – They do not know the other partial keys held by other verifier/s since the threshold verifiers are always more than one. In addition, they do not know the location of the EHR and the encrypted secret key, thus this pair will not succeed.

(ii) Unauthorized DR with DC – They do not know the other partial keys held by verifier/s and the encrypted

secret key, thus this pair will not succeed.

Threat: unauthorized DR with multiple ($\geq t$) verifiers and DC– This is the only likely collusion threat that involves the needed threshold verifiers. However, the verifiers only know their own keys but do not know the encrypted secret key.

- **Key management:**

(i) The set of keys used for SMPC are encrypted and stored on BC with a link to DO's pseudoID and ehr-id. This removes the need to store the keys in a secure location for the participants (the verifier parties) and they can retrieve the encrypted keys from BC and extract the keys using their private keys. The availability and redundancy of the BC network further facilitate key retrieval.

(ii) DO needs only to store the secret key used to encrypt the EHR and optionally, the encrypted nonce using the required set of keys. Thus, the DO's digital wallet needs to be secure, e.g. through device hardening, to protect the secret keys. The redundancy, immutability and tamper-proof properties of BC can assuage the risks of key loss, and accessibility of the encrypted keys on BC. If the set of keys is compromised and revealed, DO can just regenerate a new set of keys and provide it on BC. In the worst scenario, the EHR needs to be re-encrypted with a new secret key, and DO can re-initiate the whole process.

(iii) To revoke the keys assigned to the verifier party, DO can create a revocation list on BC to nullify the earlier record with the pseudoID and ehr-id links. If the threshold verifiers are not affected, i.e., DR still has t verifiers choices, there is no need for DO to re-initiate the whole key generation, selection of verifiers, VC creation, and key issuance processes, to provide authorized DR a new VC as well as provide the updates on BC.

7.2 Privacy analysis

A privacy analysis is similarly performed using LINDDUN [16]. We discuss the security threats and evaluate how our solution addresses them.

- **Linkability** (able to link items of interest to know the identity of the data subject(s) involved):

(i) Only DO's pseudoID is used on BC and there is no link between it and DO's identity (DID). The only link is found on the VC which is stored internally - this link is required to identify DO's ehr-id block on BC.

(ii) The EHR is not publicly available and only DC can grant its access. In addition, DC only knows the ehr-id and not the content of the encrypted EHR, thus there is no linkage to the ownership of the EHR.

- **Identifiability** (able to identify a data subject from a set of data subjects through an item of interest):

(i) PseudoID on BC is not able to identify who the DO is. The pseudoID is generated new for every VC, thus

indistinguishable for any association. The pseudoID is the only source of identifying the DO but is only recorded and identifiable on the VC.

(ii) There is no content on DC that can identify who the DO is.

- **Non-repudiation** (from the data owner's perspective of being able to deny a claim):

(i) The VC stores all required details to identify DO and its signature for her/his authorization for DRs.

(ii) Only DR knows how to derive the secret key to decrypt the data downloaded from the link provided by DC.

(iii) The interactions of DR with verifiers are also recorded on BC for reference.

- **Detectability** (able to distinguish whether an item of interest about a data subject exists, regardless of being able to read the contents itself):

(i) With only pseudoID and ehr-id recorded on BC, there is no trace to detect whose EHR it is but only a transaction record with two ids and indistinguishable from other records. The pseudoID will not be the same for the same DO.

(ii) The EHR is not publicly available and only DC can grant its access. In addition, DC only knows the ehr-id and not the content of the encrypted EHR.

- **Disclosure of Information:**

(i) BC only stores pseudoID, ehr-id and encrypted keys with no other details.

Threat: Only risk is DR's disclosure of information after decrypting EHR – this is beyond any control since the EHR is already in clear.

- **Unawareness** (unaware of the actions done on the one's (data subject) personal data):

(i) DR needs to request verification of authorization from the verifier (Notary) in order to access EHR. The verifier can optionally notify DO of any access to her/his data.

Threat: The verifier may choose not to notify DO. As the solution is on assumption that DO is unavailable to grant access, DO can detect the access only via the transaction log recorded on BC. This threat is mitigated since the EHR is encrypted and only the DO and the HSP that created it knows the secret key. For the authorized DR that eventually accesses DO's EHR, it is beyond any control since the EHR is already in clear and in DR's possession.

- **Non-compliance** (action done on personal data that is not compliant with legislation, regulation, and/or policy):

(i) As a participant in this healthcare ecosystem, each participant will have to accept the terms and conditions of use, and abide by the medical code of ethics, policies, laws, and regulations specific to its country.

Conformance to the laws and acts (e.g. HIPAA) need to be further validated.

To further analyse the disclosure of information and the information available to each participant and non-participant, we analyse the information available to each participant and present it in a 'who knows what' table as shown in TABLE 2. This provides further analysis of the privacy protection of our approach and validates whether the information available to the participants is on a need-to-know basis. Utilizing the selective disclosure of VC, DR need not provide the encrypted secret key, *cipherSK*, to the verifiers (Notary or DC) when requesting them to verify the VC. It is also not required for them in order to derive the *partialCipher* since only the nonce *r* is needed. DO's *pseudoID* used on BC will not be linkable to DO's DID unless the parties are the selected *t* verifiers since the partial keys are recorded on BC and DO's DID need to be verified by them to link the DID with the *pseudoID*.

The interactions between parties are also on an 'as-required' basis. DO is required to interact with HSP to locate its EHR in DC and with DRs when issuing the VCs to them. DR will need to interact with DO to receive the VC, with the verifiers (Notary and DC) for verification of the VC and receive the $partialCipher_{i,}$ and with DC (CSP) to get the link to download the encrypted EHR.

TABLE 2

WHO KNOWS WHAT

	DO	HSP	DR	Notary Verifier (Trusted)	DC (semi-honest)	Public	Remarks
Secret key to encrypt EHR	✓	✓					Only DO and HSP
**DO DID	✓	✓	✓	✓	✓	✓	On BC
DO pseudoID	✓	✓	✓	✓	✓	✓	On BC
DO's DID-pseudoID link	✓		✓	✓	✓		On DR's VC
cipherSK, hashedSK, keyAlloc, verifier-ids, t-verifiers	✓		✓				On DR's VC
ehr-id, encrypted keys	✓	✓	✓	✓	✓	✓	On BC
Who is Notary	✓		✓	✓	✓		On VC
Who is DC	✓	✓	✓		✓		On VC
Who are on DO's authorisation list	✓		✓	✓			On VC: Only DR's own, DO will have full list
EHR storage location/link					✓		
DO's keys to generate cipherSK	✓			✓*	✓*		*Partial only

**All DIDs are on blockchain to serve as identity proof with its public keys in a DID Document

8 CONCLUSION AND FUTURE WORK

In this chapter, we proposed an efficient, secure and privacy-protecting protocol, whereby the patient (as the data owner) has control over the pre-delegation of authorization for the access to her/his personal EHR when s/he is unavailable to grant the access. Our patient(user)-centric proposal combined Self-Sovereign Identity (SSI) concepts and framework with Secure Multi-party Computation (SMPC) and Threshold Cryptography (TC) to enable secure identity and authorization verification. Our work utilized SSI, particularly Verifiable Credentials and Decentralized Identifier, to grant authorization using SMPC for decentralized verification and access to EHR. For the threshold SMPC design and implementation, we adopted the block cipher encryption sharing approach and extended the AES-GCM symmetric encryption model into a full-fledged cryptographic platform to integrate

the block cipher encryption approach. We implemented two mechanisms for the block cipher encryption sharing, namely XOR and Cascade, and conducted experiments to compare them. We adopted the XOR mechanism as it can scale for larger thresholds, though Cascade performed better for a lower threshold (≤ 3). Through the SMPC implementation, we verified the correctness of the cryptographic operations and algorithms. Though the SSI and digital wallet building blocks were not implemented, based on the functional design of our system, we were still able to derive that the system is of polynomial time complexity whereby the cryptographic operations on the SMPC platform are the main contributor. Finally, we conducted a threat analysis on the proposed protocol and analysed its security and privacy protection to conclude that it is efficient and meets our defined requirements to provide autonomy to the patient, (as the data owner of her/his personal EHR) to control the access to her/his EHR with trusted decentralized verification of her/his pre-delegated authorization and lastly, the secure and privacy-protected access, and sharing of her/his EHR. In addition, with data encryption and secure channel (via TLS) built-in in our proposed protocol, the design is also able to meet the requirements of secure EHR sharing and access over the wireless environment networks, especially in an emergency whereby an ambulance service is required. To the best of our knowledge, there is currently no related work on the approach we have taken, thus no comparison could be made.

As an ongoing effort, we are exploring several extensions to the work in this paper:

- **Alternative methods to improve the existing SMPC-AES-GCM platform**. Further optimizing improvement and approaches can be explored for this platform. Currently, the key shares are statically recorded on the blockchain regardless of any verifier 'dropped-out', blacklisted or compromised. For more resilience to future share losses, ways to continuously refresh the key shares can be explored. In the key contribution during the block cipher encryption, different approaches can be explored – for instance, equal shares among all the t verifiers or trusted-based contribution for the most trusted (or reliable) verifier to contribute the most keys.
- **Alternative Problem Domains**. The work in this paper not only addresses the afore-mentioned challenges in the healthcare domain but also suggests an opportunity for the protocol to be extended to other domains and applications, e.g. in Power of Attorney (PoA) or Lasting Power of Attorney (LPA), proxy collection of good/item (like medicine, voucher, committee voting, etc) on behalf of the rightful recipient/member.

[**The datasets generated during and/or analysed during the current study are available from the corresponding author on reasonable request**.]

STATEMENTS AND DECLARATIONS

The authors have no competing interests to declare that are relevant to the content of this article

ACKNOWLEDGEMENT

This work was supported by the National Research Foundation, Singapore under its Strategic Capability Research Centres Funding Initiative.

REFERENCES

[1] J. Zhang, H. Liu and L. Ni, “A secure energy-saving communication and encrypted storage model based on RC4 for EHR”. *IEEE Access*, vol. 8, pp.38995-39012, 2020.
[2] X. Liu, K.Y. Lam, F. Li, J. Zhao, L. Wang and T.S. Durrani, “Spectrum sharing for 6G integrated satellite-terrestrial communication networks based on NOMA and CR”, *IEEE Network Magazine*, vol. 35, issue 4, pp. 28-34, 2021.
[3] F. Li, K.Y. Lam, H.W. Chen, N. Zhao, “Spectral efficiency enhancement in satellite mobile communications: A game-theoretical approach”, *IEEE Wireless Communications*, 2019.
[4] F. Li, K.Y. Lam, N. Zhao, X. Liu, K. Zhao, and L. Wang, “Spectrum Trading for Satellite Communication Systems with Dynamic Bargaining,” *IEEE Transactions on Communications*, vol. 66, issue 10, pp. 4680–4693, 2018.
[5] L. Wang, K.Y. Lam, M. Xiong, F. Li, X. Liu and J. Wang, “Spectrum pricing for cognitive radio networks with user's stochastic distribution”, *Wireless Networks, Springer US*, 2018.
[6] F. Li, K.Y. Lam and L. Wang. “Power allocation in cognitive radio networks over Rayleigh-fading channels with hybrid intelligent algorithms”, *Wireless Networks*, vol. 24, issue 7, pp. 2397–2407, 2018.
[7] M. Z. Chowdhury, M. Shahjalal, S. Ahmed, and Y. M. Jang, “6G wireless communication systems: Applications, requirements, technologies, challenges, and research directions,” *IEEE Open Journal of the Communications Society*, vol. 1, pp. 957–975, 2020.
[8] K. Rabieh, K. Akkaya, U. Karabiyik, and J. Qamruddin, “A secure and cloud-based medical records access scheme for on-road emergencies,” *In Proceedings of 15th IEEE Annual Consumer Communications & Networking Conference (CCNC),* pp. 1-8, 2018.
[9] X. Yi, A. Bouguettaya, D. Georgakopoulos, A. Song and J. Willemson, "Privacy protection for wireless medical sensor data," *IEEE Transactions on Dependable and Secure Computing*, vol. 13, no. 3, pp. 369-380, 2016.
[10] S. Arsheen, and K. Ahmad, "SLR: A systematic literature review on blockchain applications in healthcare," *In Proceedings of 2021 International Conference on Information Science and Communications Technologies (ICISCT)*, pp. 1-6, 2021.
[11] K.-L. Tan, C.-H. Chi, and K.-Y. Lam, “Analysis of digital sovereignty and identity: From digitization to digitalization,” *arXiv preprint arXiv:2202.10069*, 2022.
[12] F. Wang, and P. De Filippi, “Self-sovereign identity in a globalized world: Credentials-based identity systems as a driver for economic inclusion,” *Frontiers in Blockchain*, vol. 2, 2020.
[13] X.-B. Zhao, K.-Y. Lam, S.-L. Chung, M. Gu, and J.-G. Sun, "Authorization mechanisms for virtual organizations in distributed computing systems," *In Proceedings of Australasian Conference on Information Security and Privacy*, pp. 414-426, 2004.
[14] J.-P. Yong, K.-Y. Lam, S.-L. Chung, M. Gu, and J.-G. Sun, "Enhancing the scalability of the community authorization service for virtual organizations," *In Proceedings of Advanced Workshop on Content Computing*, pp. 182-193, 2004.
[15] Microsoft, "The STRIDE Threat Model," https://docs.microsoft.com/en-us/previous-versions/commerce-server/ee823878(v=cs.20)?redirectedfrom=MSDN. 2021
[16] D. R. Group, "LINDDUN privacy engineering," https://www.linddun.org/. 2022
[17] H. Guo, W. Li, M. Nejad, and C.-C. Shen, "Access control for electronic health records with hybrid blockchain-edge architecture," *In Proceedings of 2019 IEEE International Conference on Blockchain (Blockchain)*, pp. 44-51, 2019.
[18] X. Yue, H. Wang, D. Jin, M. Li, and W. Jiang, “Healthcare data gateways: found healthcare intelligence on blockchain with novel privacy risk control,” *Journal of medical systems,* vol. 40, no. 10, pp. 1-8, 2016.
[19] X. Liang, J. Zhao, S. Shetty, J. Liu, and D. Li, "Integrating blockchain for data sharing and collaboration in mobile healthcare applications," *In Proceedings of 2017 IEEE 28th annual international symposium on personal, indoor, and mobile radio communications (PIMRC)*, pp. 1-5, 2017.
[20] Q. Xia, E. B. Sifah, K. O. Asamoah, J. Gao, X. Du, and M. Guizani, “MeDShare: Trust-less medical data sharing among cloud service providers via blockchain,” *IEEE Access*, vol. 5, pp. 14757-14767, 2017.

[21] K. Rabieh, K. Akkaya, U. Karabiyik, and J. Qamruddin, "A secure and cloud-based medical records access scheme for on-road emergencies, " *In Proceedings of 2018 15th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, pp. 1-8, 2018.
[22] J. Zhang, N. Xue, and X. Huang, "A secure system for pervasive social network-based healthcare," *IEEE Access*, vol. 4, pp. 9239-9250, 2016.
[23] A. Dubovitskaya, P. Novotny, Z. Xu, and F. Wang, "Applications of blockchain technology for data-sharing in oncology: results from a systematic literature review," *Oncology*, vol. 98, no. 6, pp. 403-411, 2020.
[24] H. Guo, W. Li, E. Meamari, C.-C. Shen, and M. Nejad, "Attribute-based multi-signature and encryption for EHR management: A blockchain-based solution, " *In Proceedings of 2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pp. 1-5, 2020.
[25] Y. Sun, R. Zhang, X. Wang, K. Gao, and L. Liu, "A decentralizing attribute-based signature for healthcare blockchain, " *In Proceedings of 2018 27th International conference on computer communication and networks (ICCCN)*, pp. 1-9, 2018.
[26] J. Vora, P. Italiya, S. Tanwar, S. Tyagi, N. Kumar, M. S. Obaidat, and K.-F. Hsiao, "Ensuring privacy and security in e-health records, " *In Proceedings of 2018 International conference on computer, information and telecommunication systems (CITS)*, pp. 1-5, 2018.
[27] R. Guo, H. Shi, Q. Zhao, and D. Zheng, "Secure attribute-based signature scheme with multiple authorities for blockchain in electronic health records systems," *IEEE Access*, vol. 6, pp. 11676-11686, 2018.
[28] H. Wang, and Y. Song, "Secure cloud-based EHR system using attribute-based cryptosystem and blockchain," *Journal of medical systems*, vol. 42, no. 8, pp. 1-9, 2018.
[29] L. Chen, W.-K. Lee, C.-C. Chang, K.-K. R. Choo, and N. Zhang, "Blockchain based searchable encryption for electronic health record sharing," *Future Generation Computer Systems*, vol. 95, pp. 420-429, 2019.
[30] Y. Wang, A. Zhang, P. Zhang, and H. Wang, "Cloud-assisted EHR sharing with security and privacy preservation via consortium blockchain," *IEEE Access*, vol. 7, pp. 136704-136719, 2019.
[31] J. Benaloh, M. Chase, E. Horvitz, and K. Lauter, "Patient controlled encryption: ensuring privacy of electronic medical records, " *In Proceedings of Proceedings of the 2009 ACM workshop on Cloud computing security*, pp. 103-114, 2009.
[32] X. Liu, Y. Zheng, X. Yuan, and X. Yi, "Towards secure and lightweight deep learning as a medical diagnostic service." *In European Symposium on Research in Computer Security*, pp. 519-541. Springer, Cham, 2021.
[33] X. Liu, and X. Yi, "Privacy-preserving collaborative medical time series analysis based on dynamic time warping." *In European Symposium on Research in Computer Security*, pp. 439-460. Springer, Cham, 2019.
[34] M. Marwan, A. Kartit and H. Ouahmane, "Applying secure multi-party computation to improve collaboration in healthcare cloud," *In Proceedings of Third International Conference on Systems of Collaboration (SysCo)*, 2016, pp. 1-6, 2016.
[35] B. Shen, J. Guo, and Y. Yang, "MedChain: Efficient healthcare data sharing via blockchain," *Applied Sciences*, vol. 9, no. 6, pp. 1207, 2019.
[36] Y. Zhuang, L. R. Sheets, Y.-W. Chen, Z.-Y. Shae, J. J. Tsai, and C.-R. Shyu, "A patient-centric health information exchange framework using blockchain technology," *IEEE Journal of Biomedical and Health Informatics*, vol. 24, no. 8, pp. 2169-2176, 2020.
[37] B. Blobel, "Authorisation and access control for electronic health record systems," *International Journal of Medical Informatics*, vol. 73, no. 3, pp. 251-257, 2004.
[38] M. F. F. Khan, and K. Sakamura, "A distributed approach to delegation of access rights for electronic health records, " *In Proceedings of 2020 International Conference on Electronics, Information, and Communication (ICEIC)*, pp. 1-6, 2020.
[39] M. Joshi, K. P. Joshi, and T. Finin, "Delegated authorization framework for EHR services using attribute based encryption," *IEEE Transactions on Services Computing*, 2019.
[40] X. Liang, S. Shetty, J. Zhao, D. Bowden, D. Li, and J. Liu, "Towards decentralized accountability and self-sovereignty in healthcare systems, " *In Proceedings of International Conference on Information and Communications Security*, pp. 387-398, 2017.
[41] A. Siqueira, A. F. Da Conceição, and V. Rocha, "Blockchains and self-sovereign identities applied to healthcare solutions: A systematic review," *arXiv preprint arXiv:2104.12298*, 2021.
[42] W.-W. W. Consortium, "W3C DID Primer for Introduction, " https://github.com/w3c-ccg/did-primer. 2021
[43] M. Ge, and K.-Y. Lam, "Self-initialized distributed certificate authority for mobile ad hoc network, " *In Proceedings of International Conference on Information Security and Assurance*, pp. 392-401, 2009.
[44] P. Dunphy, and F. A. Petitcolas, "A first look at identity management schemes on the blockchain," *IEEE Security & Privacy*, vol. 16, no. 4, pp. 20-29, 2018.
[45] C. Crépeau, J. v. d. Graaf, and A. Tapp, "Committed oblivious transfer and private multi-party computation, " *In Proceedings of Annual International Cryptology Conference,* pp. 110-123, 1995.
[46] X. Yi, R. Paulet, and E. Bertino, "Homomorphic encryption," *Homomorphic Encryption and Applications*, pp. 27-46: Springer, 2014.

[47] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612-613, 1979.
[48] K. M. Martin, R. Safavi-Naini, H. Wang, and P. R. Wild, "Distributing the encryption and decryption of a block cipher," *Designs, Codes and Cryptography*, vol. 36, no. 3, pp. 263-287, 2005.
[49] S. V. Sudarsan, O. Schelén, and U. Bodin, "A model for signatories in cyber-physical systems," *In Proceedings of 2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pp. 15-21, 2020.
[50] D. Nuñez, I. Agudo, and J. Lopez, "Proxy re-encryption: Analysis of constructions and its application to secure access delegation," *Journal of Network and Computer Applications*, vol. 87, pp. 193-209, 2017.
[51] D. McGrew, and J. Viega, "The Galois/counter mode of operation (GCM)," *NIST Modes of Operation Process*, vol. 20, pp. 0278-0070, 2004.
[52] L. T. Brandão, N. Mouha, and A. Vassilev, "Threshold schemes for cryptographic primitives: challenges and opportunities in standardization and validation of threshold cryptography," *National Institute of Standards and Technology*, 2018.
[53] E. Brickell, G. D. Crescenzo, and Y. Frankel, "Sharing block ciphers," *In Proceedings of Australasian Conference on Information Security and Privacy*, pp. 457-470, 2000.
[54] U. M. Maurer, and J. L. Massey, "Cascade ciphers: The importance of being first," *Journal of Cryptology*, vol. 6, no. 1, pp. 55-61, 1993.