Blockchain Support for Flexible Queries with Granular Access Control to Electronic Medical Records (EMR)

Xiaoshuai Zhang, Stefan Poslad
School of Electrical Engineering and Computer Science
Queen Mary University of London, London, E1 4NS, United Kingdom
Email: {xiaoshuai.zhang, stefan.poslad}@qmul.ac.uk

***Abstract*—In this paper, we propose an architecture for Blockchain-based Electronic Medical Records (EMRs) called GAA-FQ (Granular Access Authorisation supporting Flexible Queries) that comprises an access model and an access authorisation scheme. Unlike existing Blockchain schemes, our access model can authorise different levels of granularity of authorisation, whilst maintaining compatibility with the underlying Blockchain data structure. Furthermore, the authorisation, encryption, and decryption algorithms proposed in the GAA-FQ scheme dispense with the need to use a public key infrastructure (PKI) and hence improve the computation performance needed to support more granular and distributed, yet authorised, EMR data queries. We validated the computation performance and transmission efficiency for GAA-FQ using a simulation of GAA-FQ against an access control scheme for EMRs called ESPAC as our baseline that was not designed using a Blockchain. To the best of our knowledge, GAA-FQ is the first Blockchain-oriented access authorisation scheme with granular access control, supporting flexible data queries, that has been proposed for secure EMR information management.**

***Index Terms*—privacy protection, access authorisation, EMR management, Internet of Things.**

I. Introduction

The global eHealth or Health Informatics market is expected to be over 300 billion dollars (in N. America) by 2022 [1]. Electronic medical records (EMRs) are a key element of this. EMRs tend to be highly distributed in terms of who (local doctor, hospital doctor, administrator etc.) has modified what and where (e.g., in hospital, doctor surgery, care homes, private homes etc.) this is done. For example, driven in part by the use of Internet of Things (IoT), wearable eHealth devices can be used outside healthcare centres, enabling not just health providers, but also health users, to monitor their own health status anywhere and anytime [2, 3]. Management of EMRs including secure storage and access control, is a crucial requirement for eHealth [4], yet is very challenging to achieve because of the highly distributed and fragmented nature of EMRs and the range of providers and users who are authorised to access them.

The use of a Blockchain model is currently being investigated as a highly distributed data structure for EMR transactions (to store, query and share) that enables them to be verified and recorded through a consensus of all parties involved [5]. Blockchain inherently enables data integrity i.e. inherently resistant to modification of the data [6]. One of the key challenges here for EMR's use of a Blockchain is that the inherent focus of a Blockchain is not to limit unauthorised, granular access to avoid specific confidential parts of an EMR implemented using a Blockchain.

As EMRs hold personal data about patients that can be confidential in different ways to different stakeholders, approaches to construct flexible and granular access authorisation to eHealth records is needed. However, the block fundamentally does not offer an appropriate level of granularity for queries and authorisation in some eHealth scenarios. For example, if a disease analyst tries to determine the incidence of different diseases in an EMR database, the data obtained by the analyst should only reveal the disease name in each block. When a nurse wants to check the drug injection doses of several patients, the information revealed to the nurse should only include patient names, IDs and drug dosage. But under the current access granularity (block), other private data of patients, e.g. other data such as social security numbers and home addresses may be present in the block, which are not related to the work of the analyst and the nurse, yet can be disclosed. This challenge motivates us to present a new EMR access architecture to achieve a more precise granularity, and flexibility, for queries and access authorisation.

Related Work. Research regarding the security of applying Blockchain is currently still in its infancy in the eHealth field. [7] proposes an access architecture, Healthcare Data Gateways (HDG) for Blockchain use in eHealth, however, its access granularity is based upon blocks. It cannot support data queries to specific data attributes in blocks or restrict the access authorisation to these attributes. [8] discusses permission controls to a Blockchain for implementing secure EMRs in different use case scenarios. A simple system structure for applying Blockchain in eHealth is presented, but the granularity of the permission control for queries is not considered. Furthermore, there are no detailed schemes or algorithms proposed. [9] implements consent management in eHealth using a Blockchain but there is no exhaustive authorisation design.

There have been various attempts to address the authorisation requirement for EMR in eHealth. [10] constructs an access control scheme, called ESPAC, to implement granularity authorisation for data queries, based upon attribute-

978-1-5386-3180-5/18/$31.00 ©2018 IEEE

based encryption (ABE) in eHealth. However, ESPAC is not Blockchain-oriented, i.e. the applied access authorisation in the scheme is for a conventional data storage structure (grid structure). Accessing one attribute of all patients requires the permission of every other patient there. Therefore, queries for one or several attributes cannot be authorised independently. Moreover, the method of public key encryption used in ABE is time-consuming as it involves bilinear pairing [11]. This is too heavy-weight to be supported in more resource-constrained eHealth IoT devices. [12] presents an access control scheme for eHealth based upon elliptic curve cryptography (ECC) but there is no support to control the access granularity in the proposed authorisation process.

Our Contribution. Compared with the existing access authorisation schemes in eHealth [7, 8, 10, 12], we highlight three novel contributions. First, to the best of our knowledge, our scheme is the first Blockchain-oriented authorisation scheme with granularity control. Second, compared with the schemes in [7, 10], the granularity of authorisation in our scheme can support different types of queries (on blocks, values of attributes or both). Third, the authorisation, encryption and decryption in our scheme does need not to rely on the public key encryption or public key infrastructure (PKI), this lowering the computation time needed.

Organisation. The remainder of the paper is organised as follows. Preliminaries is used to understand our architecture, i.e., the general Blockchain and Shamir's Secret Sharing scheme are summarised in Section II. Then, our proposed access model is introduced in Section III. After that, we discuss the security model of our authorisation scheme in Section IV. Section V describes the performance simulation of the scheme, which is followed by the final Section VI to offer the conclusions of our work.

II. Preliminaries

A. General Blockchain

A general Blockchain consists of many blocks [13]. Each block contains data, a cryptographic hash value (*h*) and a timestamp (*ts*), and the data can contain several attributes and their values. The hash value of one block is produced by the previous hash value and the data in current block. It means that the hash value is applied for establishing the link between two blocks and any fallacious and distorted data would be figured out by verifying the hash value. A general Blockchain is demonstrated as follows (Figure. 1).

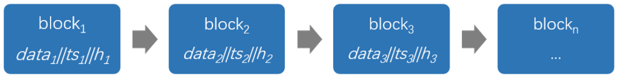


Fig. 1: The general Blockchain model

In this model, the hash values are generated by two rules:

$$h_1 = Hash(data_1||ts_1)$$
$$h_i = Hash(h_{i-1}||data_i||ts_i), i = 2...n$$

B. Shamir's Secret Sharing Scheme

Shamir's secret sharing is a common method for realising data access control. In this scheme,a secret y is divided into n shares and shared among n shareholders. If any t or more than t shares are given, it is able to reconstruct the secret y, but with fewer than t shares, it cannot reconstruct the secret.

Shamir's secret sharing scheme (SSharing) is based on Lagrange interpolating polynomials. There are n shareholders $\mathcal{U} = \{U_1, ..., U_n\}$ and a large random prime q. The scheme consists of the following algorithms:

- *SSharing.Generation(q, y)*: This algorithm takes the prime q and the secret $y \in \mathbb{Z}_q$ and does the following:

1. Pick a polynomial $f(x)$ of degree $t-1$ randomly: $f(x) = a_0 + a_1 x + ... + a_{t-1} x_{t-1}$ $(mod\ p)$, where the secret $y = a_0 = f(0)$ and all coefficients $a_0, a_1, ..., a_{t-1}$ are random in $\mathbb{Z}_q$.

2. Compute all shares: $y_i = f(x_i)$ $(mod\ q)$ for $i = 1, ..., n$, where $x_i \in \mathbb{Z}_q$ are picked randomly.

3. Output a list of n points $\{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$. Each share y_i is distributed to corresponding shareholder U_i privately. Note that x_i need not be kept secretly.

- *SSharing.Reconstruction(q,(x_{i_1},y_{i_1}),...,(x_{i_t},y_{i_t}))*: This algorithm takes the prime q and any t points $\{(x_{i_1}, y_{i_1}), ..., (x_{i_t}, y_{i_t})\}$ as inputs, it reconstructs and outputs the secret y as

$$y = f(0) = \sum_{i \in A} \Delta_i y_i \ (mod\ q),$$

where $\Delta_i = \prod_{j \in A/\{i\}} \frac{x_j}{x_j - x_i}$ $(mod\ q)$ are the Lagrange interpolation coefficients and $A = \{i_1, ..., i_t\} \subseteq \{1, ..., n\}$.

III. Proposed Access Model

There are three layers in the proposed model (Figure. 2): user layer, agent layer and storage layer. The agent layer and storage layer are located in the same trusted cloud. Next, we introduce the three layers of our access model in Figure. 2 and illustrate the functions of each layer.

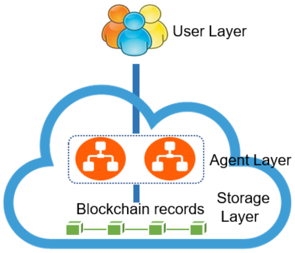


Fig. 2: The proposed access model

1) User layer: This layer represents data inquirers e.g. doctors, patients, data analysts and even monitoring devices of health. The data inquirers initiate queries to obtain data from the storage layer. All the queries will be processed through the agent layer. Note that doctors and patients normally query one or several blocks; while data analysts usually query values of different attributes (columns) in blocks to analyse morbidity and the incidence of the population. Therefore, we define three different levels of granularity for queries for different users: (i) *block query*, users request one or certain blocks on a chain;

(ii) *attribute query*, users only request all the data of particular attributes; (iii) *mixed query*: users aim to obtain the data of particular attributes from specified blocks. An example for demonstrating the three defined granular levels of queries in a Blockchain is shown in Figure. 3.

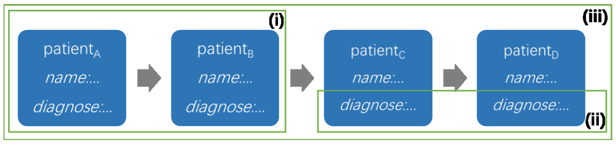


Fig. 3: The example of three levels of granularity for queries

2) Agent layer: There are two major aims of the agent layer. The first aim is to aggregate the queried data since the data could be entire blocks or columns from blocks. The second aim is to check the inquirers' access permission for the queried data and to authorise the access from the valid inquirers. In this proposed model, the agent layer can check the access permission for every block and attribute. If the inquirers have all the access permission for the queried blocks and attributes, the agent layer can authorise the access and then return the requested data. Otherwise, the agent layer should deny the access request. Note that the style of the returned data is determined by the granularity of the queries. It can be many blocks or only the data of several attributes in certain blocks.

3) Storage layer: In this layer, all the medical records for eHealth are stored in a Blockchain. Another function of this layer is to provide the queried data to the agent layer.

IV. Proposed Authorisation Scheme

In this section, we first propose our authorisation scheme, GAA-FQ (Granular Access Authorisation supporting Flexible Queries) and then prove its correctness. After that, the security of our proposed scheme is analysed.

A. The Proposed Scheme

1) ***Setup****(λ):* This procedure outputs public parameters pp with the security parameter λ using the following steps.

1. Generate a big prime q.

2. Generate a random integer token $BT \in \mathbb{Z}_q$ for each block B on the Blockchain and write BT in B. Note this step should be executed when a new block is added to the Blockchain to keep the integrity of the whole Block chain.

3. Generate a random integer token $AT \in \mathbb{Z}_q$ for each attribute A included in the blocks of the Blockchain. All (A, AT) pairs can be stored in a new Blockchain BC_{AT} separately. Note we assume that all the valid users have acquired the corresponding tokens they should have before they request data from the Blockchain.

4. Select one secure cryptographic hash function $H : \{0,1\}^* \rightarrow \{0,1\}^\lambda$

5. Select a symmetric encryption algorithm, e.g., AES (Advanced Encryption Standard) [14].

6. Output $pp = (q, BC_{AT}, H, AES)$

2) ***Query****(pp):* The user (data inquirer) prepares the query Q via following steps.

1. Choose the type of the query T from the defined three granularity queries. Note that for illustrating the remaining parts of the proposed scheme clearly we assume the type of the query is (i) *block query* and the amount of queried blocks is $n \geqslant 1$.

2. Prepare the sequence S_1 including all the identities of requested blocks. For example, S_1 should include all the identities of blocks n: $S_1 = \{B_{id_i} | i = 1...n\}$ based upon our prior assumption.

3. Prepare the sequence S_2 including all the tokens of requested blocks. For the given sequence S_1 in step 2, $S_2 = \{BT_i | i = 1...n\}$.

4. Calculate the hash value H_T of S_2 via $H_T = H(BT_1, BT_2, ..., BT_n)$.

5. Send the query $Q = (S_1, H_T)$ to the agent layer.

3) ***Authorise****(pp, Q):* The agent layer validates the access permission in Q from the user via the following steps.

1. Repeat step 3 in *Query* with $S_1 \in Q$ to obtain $S_2' = \{BT_i' | i = 1...n\}$ from the storage layer.

2. Calculate the hash value H_T' of S_2' via $H_T' = H(BT_1', BT_2', ..., BT_n')$.

3. If $H_T = H_T'$ holds, it means the user can be authorised to access the queried data and the next phases are conducted, otherwise, the agent layer should deny the access request.

4) ***Encrypt****(pp, Q, S_2'):* The agent layer encrypts the queried data via the following steps.

1. Acquire the queried data M based upon $S_1 \in Q$ from the storage layer then calculate the hash value H_M of the data M: $H_M = H(M)$.

2. Generate a secure key $k \in \mathbb{Z}_q$.

3. Use AES to encrypt M with key k to get the ciphertext $C = AES_k(M, H_M)$. For decrypting $AES_k(M, H_M)$ to recover the plain data M, AES_k' is defined as the decryption process: $M = AES_k'(C = AES_k(M, H_M))$.

4. Follow the *SSharing.Generation* in Section II.B to construct a polynomial $f(x)$ of degree n with k and S_2':

$$f(x) = k + BT_1'x + BT_2'x^2 + ... + BT_n'x^n \ (mod\ q).$$

5. Generate a random integer $x_p \in \mathbb{Z}_q$ and calculate a point $P(x_p, y_p = f(x_p))$.

6. Return (C, P) to the inquirer to finish the authorisation and data transmission.

5) ***Decrypt****(pp, Q, C, P):* The user has all the valid access permissions for the queried data and can decrypt the ciphertext C after the *Authorise* and the *Encrypt* phases via the following steps.

1. Use the sequence $S_2 = \{BT_i | i = 1...n\}$ organised in former *Query* phase to construct a polynomial $g(x)$ based upon the *SSharing.Generation* in Section II.B:

$$g(x) = a_0 + BT_1x + BT_2x^2 + ... + BT_nx^n \ (mod\ q).$$

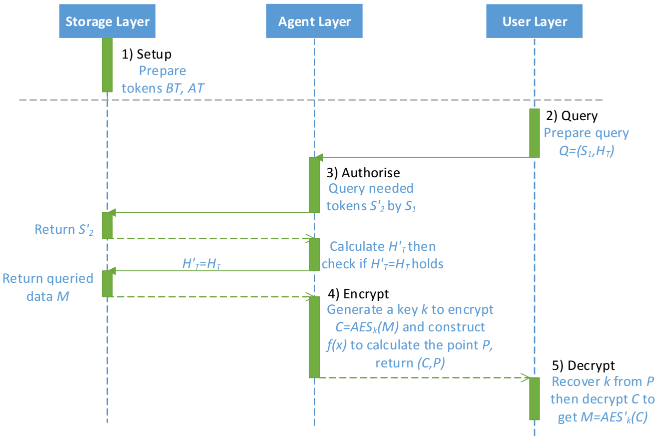
2. Follow the *SSharing.Reconstruction* in Section II.B to recover the key $k = g(0) = a_0 \in \mathbb{Z}_q$ for AES decryption with $P(x_p, y_p)$:

$$k = y_p - BT_1x_p - BT_2x_p^2 - ... - BT_nx_p^n \ (mod\ q).$$

3. Decrypt C to retrieve the plaintext $(M, H_M) = AES'_k(C) = AES'_k(AES_k(M, H_M))$.

4. If $H(M) = H_M$ holds, this algorithm outputs M; otherwise, it outputs $\perp$.

The following Figure. 4 shows the work flow of our scheme.



Storage Layer
Agent Layer
User Layer
1) Setup
Prepare tokens BT, AT
2) Query
Prepare query Q=(S1,HT)
3) Authorise
Query needed tokens S'2 by S1
Return S'2
Calculate H'T then check if H'T=HT holds
H'T=HT
Return queried data M
4) Encrypt
Generate a key k to encrypt C=AESk(M) and construct f(x) to calculate the point P, return (C,P)
5) Decrypt
Recover k from P then decrypt C to get M=AES'k(C)

Fig. 4: The work flow of GAA-FQ

B. Correctness

In the *Authorise* phase, $H_T = H'_T$ is the condition for a successful authorisation.

$$H_T = H'_T \Leftrightarrow H(S_2) = H(S'_2)$$
$$\Leftrightarrow H(\{BT_1, BT_2, ..., BT_n\}) = H(\{BT'_1, BT'_2, ..., BT'_n\})$$
$$\Leftrightarrow \{BT_1, BT_2, ..., BT_n\} = \{BT'_1, BT'_2, ..., BT'_n\}.$$

It means the user can pass the *Authorise* phase if and only if this user has all the corresponding access permission tokens for the requested blocks.

In the *Decrypt* phase, the authorised inquirer can retrieve the key $k = a_0 \in \mathbb{Z}_q$ for AES decryption based upon the *SSharing.Reconstruction*. Given a point $P(x_p, y_p)$ on the polynomial $f(x)$, it should be on the correct reconstructed polynomial as well. Because the condition $\{BT_1, BT_2, ..., BT_n\} = \{BT'_1, BT'_2, ..., BT'_n\}$ holds after the authorisation, the reconstructed polynomial $g(x)$ is same as the original polynomial $f(x)$ except for the unknown first coefficient a_0. This means determining the secret $g(0) = a_0 = k \in \mathbb{Z}_q$ requires only one point (shareholder) $P(x_p, y_p)$:

$$\begin{aligned} k &= a_0 \\ &= g(x_p) - BT_1 x_p - BT_2 x_p^2 - ... - BT_n x_p^n \\ &= y_p - BT_1 x_p - BT_2 x_p^2 - ... - BT_n x_p^n \\ &= f(x_p) - BT_1 x_p - BT_2 x_p^2 - ... - BT_n x_p^n \\ &= f(x_p) - BT'_1 x_p - BT'_2 x_p^2 - ... - BT'_n x_p^n \\ &= k \ (mod\ q). \end{aligned}$$

Hence, the authorised user can reconstruct the polynomial $g(x)$ and restore the correct key k for the *Decrypt* phase.

C. Security Analysis

In this section, we analyse the security of our authorisation scheme from two angles, data confidentiality and integrity.

1) Confidentiality: The confidentiality we focus on is between the user layer and the agent layer. There are two potential attacks may occur in the communications between the user layer and the agent layer.

(i) The user tries to access the data but without the corresponding access permission. For example, the user requests the data for several attributes $S_1 = \{A_1, A_2, ..., A_n\}$, however, this user only has the partial access tokens $\{AT_1, AT_2, ..., AT_{n-1}\}$ for $\{A_1, A_2, ..., A_{n-1}\}$. Therefore, the user forges the access token $AT_n^* \in \mathbb{Z}_q$ for the attribute A_n then constructs the fake $S_2^* = \{AT_1, AT_2, ..., AT_{n-1}, AT_n^*\}$ to calculate the hash value $H_T^* = H(S_2^*) = H(AT_1, AT_2, ..., AT_{n-1}, AT_n^*)$. Finally, the user sends the query $Q^* = (S_1, H_T^*)$ to the agent layer.

For the agent layer, the true token sequence queried from the storage layer is $S'_2 = \{AT'_1, AT'_2, ..., AT'_{n-1}, AT'_n\}$. Although $\forall i \in \{1..n-1\}, AT_i = AT'_i$ holds, the user only has the advantage $Pr[AT'_n = AT_n^*] = \frac{1}{|\mathbb{Z}_q|}$ to satisfy the condition $AT'_n = AT_n^*$, where $|\mathbb{Z}_q|$ represents the number of all the elements in $\mathbb{Z}_q$. Note that $\mathbb{Z}_q$ is a large discrete space as q is a big prime. Thus, $|\mathbb{Z}_q|$ is big enough to keep the user's advantage $Pr[AT'_n = AT_n^*] = \frac{1}{|\mathbb{Z}_q|}$ is negligible. Also, the probability of $S_2^* = S'_2$ is negligible as well. As a result, the user cannot pass the *Authorise* phase (see Section IV.A.3) since $Pr[H_T^* = H'_T] = Pr[S_2^* = S'_2] = Pr[AT'_n = AT_n^*] = \frac{1}{|\mathbb{Z}_q|}$ is negligible based upon the above analysis.

(ii) The communications between the user layer and the agent layer are eavesdropped on by the attacker. The attacker can obtain all the data from the communications between the user layer and the agent layer. Based upon our scheme, the data includes the query $Q = (S_1, H_T)$ and the response (C, P).

If the attacker modifies the query Q, it cannot pass through the *Authorise* phase in our scheme based upon the security analysis (i). On the other hand, the attacker cannot retrieve the plain data M only with Q and (C, P) because of the following two reasons. Firstly, the attacker cannot recover the correct sequence of access tokens from the hash value H_T since the secure cryptographic hash function is a one-way function. Thus, the attacker cannot determine the correct coefficients of the polynomial $g(x)$. Secondly, the proper polynomial including the point P cannot be determined since the only one point P could be on an infinite number of polynomials. As a result, the attacker cannot carry on the *SSharing.Reconstruction* or recover the key $k = a_0 \in \mathbb{Z}_q$ to decrypt ciphertext C without the correct $g(x)$.

2) Integrity: If the attacker eavesdrop upon the communications between the user layer and the agent layer, the attacker can challenge the integrity with the intercepted data (C, P) through three methods.

- *Forged* C^*: If the attacker changes C to C^*, the user can still recover the correct key k to decrypt C^*. However, based upon the AES algorithm, the decrypted $AES'_k(C^*) = (M^*, H_{M^*})$ is changed and the attacker cannot control M^* and H_{M^*} to make $H(M^*) = H_{M^*}$ hold via tampering C^*.
- *Forged* P^*: If the attacker replaces the point P to a fake point P^*, the user can only recover an incorrect key

k^* with *SSharing.Reconstruction* to decrypt C. Furthermore, based upon the AES algorithm, the decrypted $AES'_{k^*}(C) = (M^*, H_{M^*})$ is changed and the attacker cannot control M^* and H_{M^*} to make $H(M^*) = H_{M^*}$ hold via tampering k^*.

• *Forged* (C^*, P^*): The integrity analysis for this situation is a combination of the former two situations. According to the analysis for *Forged* C^* and *Forged* P^*, we conclude that the attacker cannot control M^* or H_{M^*} to make $H(M^*) = H_{M^*}$ hold via tampering C^* or P^*. Hence, step 4 of the *Decrypt* phase outputs $\perp$ to indicate the failed integrity check for the three attack methods.

Overall, when attacks happen to the communications between the user layer and the agent layer, our scheme offers sufficient security to ensure that both the confidentiality of the transported data and the data integrity is preserved based upon the security analysis.

V. Performance Simulation

There are two parts to be illustrated in this section. The first part is the theoretical comparison of cryptographic operations. In the second part, the computational time efficiency for performance of GAA-FQ is evaluated with respect to the time cost for transmitting encrypted data over Wi-Fi. The reason for evaluating the time cost of data transmission over Wi-Fi is that a lower usage time of Wi-Fi means lower power cost for the resource-constrained eHealth devices when transmitting or receiving data. Since there is yet no clear best practice to be used as a baseline for comparison, we select a granularity access authorisation scheme based upon cloud computing named ESPAC [10] as our baseline. ESPAC is also applied in eHealth but uses conventional (non-Blockchain) data structure.

A. Theoretical Comparison

The two major cryptographic operations used in ESPAC [10] are bilinear pairing and scalar multiplication. We denote by O_{pair} an operation of the bilinear pairing, and O_{mul} an operation of the scalar multiplication in ESPAC. The notation O_{exp} represents the modular exponentiation operation needed in GAA-FQ. Because the used quantity of the cryptographic operations is determined by the number of required tokens in GAA-FQ (*resp.* attributes in ESPAC), we assume the number of the tokens (*resp.* attributes) used in the authorisation of the two schemes is k for the equal-scale comparison of used operations. The result is given in Table I.

TABLE I: Theoretical comparison of used operations

	GAA-FQ	ESPAC
Encryption	kO_{exp}	$1O_{pair} + 2kO_{mul}$
Decryption	kO_{exp}	$k(2O_{pair} + O_{mul})$

Note that there are other cryptographic algorithms used in schemes GAA-FQ and ESPAC, e.g. hash summary and AES. However, time complexity of these algorithms is negligible [15] when compared to the denoted operations. Hence, these algorithms are not taken into account in the above comparison.

B. Simulation Comparison

In this part, we describe our simulation to compare the time efficiency of computation and transmission for the two schemes. All the results are averaged over 10 runs for each number of the used tokens (*resp.* attributes). The devices for our simulation use are a conventional computer with an Intel i5-4200H processor running at 3.30GHz, and a Raspberry Pi 2 as a low-resource eHealth device.

We first vary the number of the tokens used for authorisation in GAA-FQ (*resp.* attributes in ESPAC) to compare the time taken for the encryption and decryption algorithms in the two schemes on the conventional computer. The simulation is implemented using MIRACL [16] and *cpabe* toolkit [17]. They can support all the necessary symmetric-key and asymmetric-key cryptographic algorithms, and ciphertext-policy attribute-based encryption (CP-ABE). Note that the number of the tokens (*resp.* attributes) used in the authorisation varies from 2 to 10 with 10 runs for each number of tokens to acquire the averaged results. All the experiments use an equivalent cryptographic security level (128-bit security) [18].

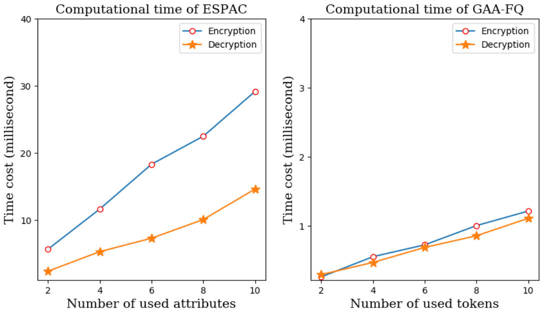


Fig. 5: The computational time cost of encryption and decryption algorithms in the schemes ESPAC and GAA-FQ on the conventional computer

The simulation result is depicted in Figure. 5. Since the bilinear pairing operations cost much more time in ESPAC, the time efficiency of our GAA-FQ is significantly superior. On average, the time consumption of GAA-FQ is only about 6% that of ESPAC in terms of the encryption and decryption algorithms. In addition, the time cost's growth rate of our encryption and decryption algorithms is lower than the relative growth rate in ESPAC by Figure. 5.

Next, we repeat the above simulation on the Raspberry Pi 2 with only changing the number of the used tokens (*resp.* attributes) to 5, 10 and 15. Meanwhile, we test the time cost of encrypted data transmission over Wi-Fi in the two schemes. The data transmission is implemented based upon Python-2.7 socket communication. The length of the plain response data from the agent layer to the user layer is 256 bytes.

The simulation results in Figure. 6 show that the comparison result of the computational time cost on the Raspberry Pi 2 is consistent with the results on the conventional computer. On the other hand, for the transmission efficiency, the time cost of the encrypted data transmission over Wi-Fi increases

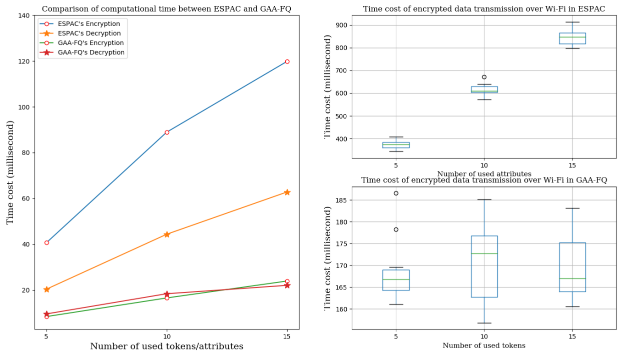


Fig. 6: The comparison of the computation and transmission time cost on the Raspberry Pi 2 for ESPAC and GAA-FQ

linearly with the number of attributes in ESPAC. However, the corresponding time cost in GAA-FQ is lower and kept stable i.e. it is non-sensitive to the number of tokens. This is because the additional contents in the encrypted data of ESPAC includes all the used attributes for the following decryption, while in GAA-FQ, the additional part is only the contents of one point regardless of the number of the used tokens.

VI. Conclusion

In this paper, we propose what we believe is the first architecture with flexible granular access authorisation to support flexible queries, GAA-FQ, for Blockchain-oriented EMR in eHealth. Compared with the existing work, e.g., ESPAC, GAA-FQ does not require PKI to authorise the access or encrypt/decrypt the queried EMRs. The proposed access authorisation scheme combined with three exemplar types of queries achieves a finer granular control when authorising different queries. As a result, a Blockchain-based EMR can respond to a requester without leaking unauthorised private data efficiently, especially for a resource-constrained device in an IoT eHealth system.

Acknowledgement

This research was funded in part by a PhD scholarship funded jointly by the China Scholarship Council and Queen Mary University of London.

References

[1] Grand View Research, “eHealth Market Analysis By Product, By Services, By End-Use And Segment Forecasts To 2022,” [Online]. Available: http://www.grandviewresearch.com/industry-analysis/e-health-market, 2017.

[2] N. K. Suryadevara and S. C. Mukhopadhyay, “Wireless sensor network based home monitoring system for wellness determination of elderly,” *IEEE Sensors Journal*, vol. 12, no. 6, pp. 1965–1972, 2012.

[3] W. Leister, M. Hamdi, H. Abie, A. Torjusen, and S. Poslad, “An evaluation framework for adaptive security for the iot in ehealth,” *International Journal on Advances in Security*, vol. 7, no. 3–4, pp. 93–109, 2014.

[4] M. Al Ameen, J. Liu, and K. Kwak, “Security and privacy issues in wireless sensor networks for healthcare applications,” *Journal of medical systems*, vol. 36, no. 1, pp. 93–101, 2012.

[5] A. Dubovitskaya, Z. Xu, S. Ryu, M. Schumacher, and F. Wang, “How blockchain could empower ehealth: An application for radiation oncology,” in *VLDB Workshop on Data Management and Analytics for Medicine and Healthcare*. Springer, 2017, pp. 3–6.

[6] Wikipedia, “Blockchain.” [Online]. Available: https://en.wikipedia.org/wiki/Blockchain, 2017.

[7] 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, p. 218, 2016.

[8] A. Dubovitskaya, Z. Xu, S. Ryu, M. Schumacher, and F. Wang, “Secure and trustable electronic medical records sharing using blockchain,” *arXiv preprint arXiv:1709.06528*, 2017.

[9] P. Genestier, S. Zouarhi, P. Limeux, D. Excoffier, A. Prola, S. Sandon, and J.-M. Temerson, “Blockchain for consent management in the ehealth environment: A nugget for privacy and security challenges,” *Journal of the International Society for Telemedicine and eHealth*, vol. 5, pp. 24–1, 2017.

[10] M. Barua, X. Liang, R. Lu, and X. Shen, “Espac: Enabling security and patient-centric access control for ehealth in cloud computing,” *International Journal of Security and Networks*, vol. 6, no. 2-3, pp. 67–76, 2011.

[11] R. W. Zhu, G. Yang, and D. S. Wong, “An efficient identity-based key exchange protocol with KGS forward secrecy for low-power devices,” *Theoretical Computer Science*, vol. 378, no. 2, pp. 198–207, 2007.

[12] J. Liu, Y. Xiao, and C. P. Chen, “Authentication and access control in the internet of things,” in *Distributed Computing Systems Workshops (ICDCSW), 2012 32nd International Conference on*. IEEE, 2012, pp. 588–592.

[13] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” 2008.

[14] J. Daemen and V. Rijmen, *The design of Rijndael: AES-the advanced encryption standard*. Springer Science & Business Media, 2013.

[15] T. Eisenbarth and S. Kumar, “A survey of lightweight-cryptography implementations,” *IEEE Design & Test of Computers*, vol. 24, no. 6, 2007.

[16] M. Scott, “MIRACL - Multiprecision Integer and Rational Arithmetic C/C++ Library.” [Online]. Available: https://github.com/miracl/MIRACL, 2012.

[17] J. Bethencourt, A. Sahai, and B. Waters, “Ciphertext-policy attribute-based encryption,” in *Security and Privacy, 2007. SP’07. IEEE Symposium on*. IEEE, 2007, pp. 321–334.

[18] E. Barker, W. Barker, W. Burr, W. Polk, and M. Smid, “Recommendation for key management part 1: General (revision 3),” *NIST special publication*, vol. 800, no. 57, pp. 1–147, 2012.