Patient-Centric Blockchain for Secure EHRs with Attribute-Based Encryption

Tarun Ramkumar Gargeya
Department of CSE
Amrita School of Engineering
Amrita Vishwa Vidhyapeetham
Chennai, India
tarungargeya@gmail.com

Thupalli Samireddy Divija
Department of CSE
Amrita School of Engineering
Amrita Vishwa Vidhyapeetham
Chennai, India
divijareddyts@gmail.com

Corresponding Author
Dr. J Umamageshwaran
Department of CSE
Amrita School of Engineering
Amrita Vishwa Vidhyapeetham
Chennai, India
j_umamageshwaran@ch.amrita.edu

Bilinear Maps and Assumptions

Bilinear Map: Let x and x_T be the cyclic groups of prime power p. The map is given by $e : X \times X \to X_T$ satisfies:

- **Bilinearity:** $e(x^m, x^n) = e(x, x)^{mn}$ for all $m, n \in \mathbb{Z}_p$.
- **Non-degeneracy:** $e(x, x) \neq 1$ for generator x.

Decisional Bilinear Diffie-Hellman (DBDH) Assumption [2]: Given (x, x^m, x^n, x^o, Z) where $m, n, o \in \mathbb{Z}_p$, it is computationally tough to differentiate whether $Z = e(x, x)^{mno}$ or Z is a random element in X_T.

Access Structures

Access Tree: Access Tree: Tree data structure that depicts access policy. Leaf nodes denote the attributes, and internal nodes are logical AND/OR operations.

LSSS Matrix: LSSS Matrix: The LSSS matrix M is the mathematical depiction of the access policy where each row denotes an attribute.

A. Method for Generating Traceable Electronic Health Records (EHRs)

Certificates are issued to various roles, including the Attribute Authority (AA), Data Owner (DO), and Data User (DU). Global public parameters such as the Master Secret Key (MSK), Public Key (PK), and Attribute Key (AK) are generated.

Step 1: Global Settings

Certificates are issued to various roles, including the Attribute Authority (AA), Data Owner (DO), and Data User (DU). Global public parameters such as the Master Secret Key (MSK), Public Key (PK), and Attribute Key (AK) are generated.

Setup (λ)

This is the initialization algorithm. The Certificate Authority (CA) chooses a group G of order p with generator g. Another group, G_T, is defined, where $e(g, g)$ is an element of G_T, and H_1 is a hash function. Multiple Attribute Authorities (AA_i, where $i = 1, 2, \ldots$) manage attribute applications. Each AA oversees a subset of attributes, reducing centralization and mitigating risks from untrustworthy authorities.

In the initialization, each AA_i initializes and generates an attribute version parameter V_x corresponding to the attributes it manages. This parameter helps quickly revoke attributes if needed. The AA_i sends these parameters to the CA, which calculates the attribute version parameters and derives the corresponding attribute key as $AK_x = g^{V_x}$.

The following steps occur:

- Random values $\alpha, \beta, a \in \mathbb{Z}_p$ are generated.
- Public Key: $PK = \{g, e(g, g)^\alpha, g^a\}$.
- Master Key: $MSK = \{PK, g^\alpha, \beta\}$.

The version parameter set $\{V_x\}_{x \in X}$, where X represents the attributes managed by the AA, is shared with the CA but kept secret from other roles. The CA distributes the PK and AKs to relevant roles, including AA_i, Encryption and Decryption Services (EPS), and Data Users (DU).

Step 2: Patient Appointment with Hospital

Patients (P) book an appointment with a hospital (H) for medical treatment. The process involves the following steps:

Step 2.1

The patient (P) establishes a connection with the hospital (H) and sends the encryption algorithm set I and the hash function H_1 to H. The hospital selects an algorithm E_i ($i \in I$) from the set, uses it for encryption, and returns its certificate (Cer$_H$) to the patient.

Step 2.2:

Upon receiving the certificate Cer$_H$, its validity is checked. This involves verifying whether the certificate falls within its validity period and ensuring the issuer is legitimate. Upon successful validation, P randomly selects $a \in \mathbb{Z}_p$ and computes $A = g^a$. Then, P encrypts A with the symmetric encryption algorithm E_i, utilizing the random number u, resulting in $E_A = E_i(u, A)$. The certificate Cer$_H$ is used to encrypt u as E_U. P computes the hash value of A using H_1, resulting in hash$_A$. Finally, P forwards the message $\{E_A, E_U, \text{hash}_A\}$ to H.

Step 2.3:

Once it receives the message, H decrypts and checks the message. Using its private key, H decrypts u and then decrypts A as $A = D_i(u, E_A)$. If $H_1(A') = H_1(A)$, the verification succeeds; otherwise, it fails. Next, H generates a random number $b \in \mathbb{Z}_p$ and calculates $B = g^b$. Using u from P, H encrypts B as $E_B = E_i(u, B)$, calculates the hash value of B as hash_B using H_1, and returns $\{E_B, \text{hash}_B\}$ to P.

Step 2.4:

Once both parties receive their respective messages, P calculates $AK_P = B^a = g^{ab}$, and H calculates $AK_H = A^b = g^{ab}$. Notably, $AK = AK_P = AK_H$. The Certificate Authority (CA) then uses AK to encrypt P's appointment details AP as $E_P = E_i(AK, AP)$. H receives E_P and decrypts it to obtain AP. H assigns the appropriate doctors $\{D_j\}_{j \in J}$ to P and delivers their IDs $\{ID_{D_j}\}_{j \in J}$. The diagnosis and treatment details, including the time (TimePeriod_P), location (Location_P), and additional information (Aux_P), are encrypted with AK and then delivered to P. P decrypts the ciphertext to extract the information about the doctors along with the associated treatment details.

Step 3:

The doctor produces the relevant EHR M, which depends on the diagnosis and treatment he offers to the patient. In addition to this, steps are undertaken to track down responsibility for the diagnosis outcome. For this purpose, the doctor puts his signature to the diagnosis. To make data safe from tampering and alteration, the blockchain is used to hold information.

The hash value $B_{\text{hash},t}$ of the latest block is obtained from the current blockchain, which represents the hash value of the block at time t. The generated data DP is defined as:

$$DP = B_{\text{hash},t} \| \text{hash}(M) \| h(ID_P) \tag{2}$$

Where:

- $\text{hash}(M)$ represents the hash value of the generated EHR M,
- $h(ID_P)$ denotes the hash value of the patient's ID to ensure privacy and data integrity.

Step 1: Encrypt the EHR

The patient (P) encrypts their EHR before uploading its ciphertext.

Encryption Algorithm

This algorithm is used for encryption. The input includes:

- **Public Key** (PK): A global public key generated by the system.
- **Attribute Key** (AK_x): The secret key associated with attributes corresponding to role x.
- **Access Control Policy** ((M, ρ)):
  - M: A matrix in the Linear Secret Sharing Scheme (LSSS) access structure [2].
  - ρ: A mapping function that associates attributes to the rows of matrix M.
- **Message** (m): The EHR to be encrypted.

The algorithm outputs the ciphertext (CT).

Algorithm Details

The Data User (DU) performs the encryption. The inputs and steps are as follows:

Input Details:

- PK: The public key.
- AK_x: The attribute key for role x.
- (M, ρ):
  - M: A matrix with l rows and n columns, where each attribute in the access structure corresponds to a specific row of M.
  - ρ: A mapping function that links attributes to the rows of M.

Encryption Steps:

1) Randomly select a column vector:
$$v = (s, y_2, y_3, \ldots, y_n) \in \mathbb{Z}_p^n$$
Here, s is the secret parameter to be shared, and $y_2, y_3, \ldots, y_n$ are random values.
2) Compute:
$$\lambda_i = M_i v, \quad \forall i \in \{1, 2, \ldots, l\}$$
Where M_i is the i-th row of the matrix M.
3) Randomly generate:
$$r_1, r_2, \ldots, r_l \in \mathbb{Z}_p$$
4) Encrypt the message m as follows:
$$CT = \{C, C', C_i, D_i\}_{i=1}^{l}$$
Where:
$$\begin{aligned} C &= m \cdot e(g, g)^{\alpha s} \\ C' &= g^s \\ C_i &= g^{a\lambda_i} \cdot (g^{v_{x_i}})^{-r_i}, \quad \forall i \in \{1, 2, \ldots, l\} \\ D_i &= g^{r_i}, \quad \forall i \in \{1, 2, \ldots, l\} \end{aligned}$$

Explanation of Symbols

- $e(g, g)$: The bilinear pairing operation.
- α: A parameter associated with the master secret key.
- g: A generator of the bilinear group.
- a: A secret value associated with the attribute authority.
- v_{x_i}: The attribute key element corresponding to attribute x_i.

Step 2: Generation of Transform Key and Private Key

To enable doctors or other authorized roles to access a patient's EHR, both a transformation key (TK) and a private key (SK) are required. These keys are generated using the KeyGen algorithm and distributed accordingly: the transformation key is entrusted to the Encryption Proxy Server (EPS) for partial decryption, while the private key is provided to the Data User (DU) for full decryption.

Key Generation Algorithm

The KeyGen algorithm proceeds as follows:

- **Inputs:**
  - **MSK:** The master secret key.
  - **S:** The attribute set associated with the user.
- **Outputs:**
  - **TK:** The transformation key, is used by the EPS for partial decryption.
  - **SK:** The private key, used by the DU for complete decryption.

Transformation Key (TK) Generation

The transformation key is calculated as:

$$TK = \{K, L, \{K_x\}_{x \in S}\}$$

Where:

$$K = g^{\frac{\alpha}{z}} \cdot g^t$$

$$L = g^t$$

$$K_x = g^{V_x \cdot t}, \quad \forall x \in S$$

Parameter Definitions

The parameters used in the transformation key generation are defined as follows:

- g: A generator of the bilinear group.
- z: A random value chosen from the set $\mathbb{Z}_p$.
- $t = \frac{\beta}{z}$: A system parameter dependent on z.
- S: The set of attributes assigned to the DU.
- V_x: A value associated with the attribute x.
- α: A master secret parameter.
- β: A system-defined constant.

Each user is assigned a unique z, ensuring that the transformation key TK is specific to that user.

Private Key (SK) Generation

The private key for the Data User (DU) is constructed as:

$$SK = \{z, TK\}$$

This private key combines the unique random value z with the corresponding transformation key TK. Together, these enable the DU to fully decrypt the ciphertext after partial decryption by the EPS.

Step 3: Partial Decryption by the EPS

Given that mobile devices have limited computational power and that Attribute-Based Encryption (ABE) is computationally intensive, especially for complex access policies, the decryption process is partially outsourced to the Encryption Proxy Server (EPS). By offloading some decryption operations to the EPS, the ciphertext is transformed into an ElGamal-style ciphertext, reducing the computational burden for the Data User (DU).

Partial Decryption Algorithm: Transform(TK, CT): This algorithm performs partial decryption of the ciphertext:

- **Inputs:**
  - CT: The encrypted ciphertext.
  - TK: The transformation key provided to the EPS.
- **Output:**
  - CT': The partially decrypted ciphertext.

1) **Input Validation:** The EPS receives both TK and CT from the DU. It uses the access structure defined by the Linear Secret Sharing Scheme (LSSS) to verify if the attribute set S satisfies the access policy embedded in CT.
   - If S does not satisfy the policy, the algorithm outputs $\perp$ (failure).
   - Otherwise, the EPS calculates the necessary linear combinations of shares, $\omega_i \in \mathbb{Z}_p$, such that:

   $$\sum_{i \in I} \omega_i \lambda_i = s$$

   Where:
   - $\{\lambda_i\}$ represents the valid shares of the secret parameter s.
   - I is the set of indices that correspond to the valid attributes in the policy.
2) **Intermediate Computation:** Using the ciphertext components and the transformation key, the EPS performs the following calculation:

   $$e(C, K) \cdot \prod_{i \in I} e(C_i^{\omega_i}, L) \cdot \prod_{i \in I} e(D_i^{\omega_i}, K_{\rho(i)}) = e(g, g)^{\frac{s\alpha}{z}}$$

   Where:
   - C, C_i, D_i: Components of the ciphertext CT.
   - $K, L, K_{\rho(i)}$: Components of the transformation key TK.
   - $e(g, g)$: The bilinear pairing operation.
3) **Partial Decryption Output:** The partially decrypted ciphertext CT' is generated as:

   $$CT' = \{C, e(g, g)^{\frac{s\alpha}{z}}\}$$

Explanation of Components

- **Bilinear Pairing** $e(g, g)$**:** Used in ABE schemes to map elements from two groups to a third group while preserving the structure.
- **Transformation Key** TK**:** Contains the necessary parameters for the EPS for partial decryption.
- **Secret Parameter** s**:** A hidden value shared in the access policy is reconstructed by combining valid shares λ_i.

This approach enables the DU to complete the decryption efficiently, even on resource-constrained devices.

Step 4: Final Decryption of the EHR Ciphertext by the DU

The DU performs the final decryption of the EHR ciphertext using the Decrypt(CT, SK) algorithm. This algorithm takes the partially decrypted ciphertext CT and the DU's secret key SK as inputs, and outputs the decrypted plaintext message m, or $\perp$ if decryption fails.

The process works as follows:

1) The partially decrypted ciphertext CT is sent from the EPS to the DU. Since this ciphertext is in an ElGamal-style format, the DU uses their secret key SK to complete the decryption.
2) The plaintext message m is then computed using the following formula:

$$m = \frac{C}{e(g,g)^{\frac{s\alpha}{z}}}$$

Where:

- C is a component of the ciphertext.
- $e(g,g)^{\frac{s\alpha}{z}}$ is the result of a bilinear pairing operation, partially decrypted by the EPS.

If decryption is successful, m is the recovered plaintext message. If the decryption fails, the output is $\perp$.

Step 5: Attribute Revocation

The attribute revocation process ensures that when an attribute of the DU is revoked or added, the user is immediately denied access to EHRs in the access policy associated with the attribute. That is, there is an updating of attribute keys and transformation keys to reflect revocation.

Attribute Revocation Process: When an attribute of a DU is revoked, the Certificate Authority (CA) will instruct the Attribute Authorities (AAs) to manage the distribution and update the version of the revoked attribute. If the version corresponding to an attribute x is V_x, a new version V_x' needs to be created. The updated attribute key AK_x is computed as:

$$AK_x = g^{V_x'}$$

The updated attribute key AK_x is then sent to the CA, which transmits it to all DUs that still possess the attribute to be updated. Each AA maintains an attribute application list and generates an upgrade key UK for users who still have the attribute:

$$UK = g^{(V_x' - V_x)\cdot\frac{\beta}{z}}$$

Note that different users have different z values, preventing one user from using another user's upgrade key. The DU uses the UK to upgrade their transformation key TK, updating only the revoked attribute while leaving the others unchanged. The updated attribute key K_x is:

$$K_x = K_x \cdot UK$$

The updated transformation key TK is as follows:

$$TK = \left\{K = g^{\frac{\alpha}{z}} \cdot g^t, L = g^t, \{K_u\}_{u\neq x}, K_x = K_x \cdot UK\right\}$$

After updating the transformation key, the ciphertext CT needs to be updated, as it is also tied to the attribute key. For an attribute, the attribute update key AUK_{xi} is generated as:

$$AUK_{xi} = D^{-(V_{xi}' - V_{xi})} \cdot i$$

The update key AUK_{xi} is then sent to the Data Owner (DO). Upon receiving the update, the DO updates the ciphertext by applying this key to the affected attribute. The component C_i related to the revoked attribute is updated, while the other attributes remain unchanged. The updated ciphertext CT^* is:

$$CT^* = \left\{\begin{array}{r} C = m \cdot e(g,g)^{\alpha s}, \\ C' = g^s, \\ \{D_i\} = g^{r_i}, \\ C_i = g^{a\lambda_i} \cdot g^{V_u - r_i}, \\ C_x = C_x \cdot AUK_x \end{array}\right\}$$

References

[1] Y. Zhang, C. Xu, S. Yu, H. Li, and X. Zhang, "SCLPV: Secure certificateless public verification for cloud-based cyber-physical-social systems against malicious auditors," *IEEE Trans. Comput. Social Syst.*, vol. 2, no. 4, pp. 159–170, Dec. 2015.

[2] Y. Jiang, X. Xu and F. Xiao, "Attribute-Based Encryption With Blockchain Protection Scheme for Electronic Health Records," *IEEE Transactions on Network and Service Management*, vol. 19, no. 4, pp. 3884-3895, Dec. 2022.