DAC-PRE: Practical Anonymous Data Access Scheme Control with Proxy Re-encryption for Implantable Medical Devices

Jayaprakash Kar[1], Xiaoguang Liu[2,3], and Fagen Li[2,4]

[1]Centre for Cryptology, Cyber Security and Digital Forensics, Department of Computer Science & Engineering, The LNM Institute of Information Technology, Jaipur(Raj), India
[2]Guangxi Key Laboratory of Cryptography and Information Security, Guilin University of Electronic Technology, Guilin, Guangxi 541004, P. R. China
[3]School of Mathematics, Southwest Minzu University, Chengdu,610041, China
[4]School of Computer Science and Engineering University of Electronic Science and Technology of China Chengdu, China

***Abstract*—Data access control is one of the most essential key components of ensuring data security. The data access control is a selective restriction of data access, and it usually comprises two main security components: authentication and authorization. In this paper, we first propose an efficient data access control scheme for implantable medical devices. The protocol ensures anonymous data access control through a signcryption scheme with proxy re-encryption (DAC-PRE) and supports the anonymity behavior of the user. Then the security of the scheme is proven in the random oracle model. Finally, We analyze the proposed protocol, and the results show that our scheme has very a low cost with respect to the computation time, communication overhead, as well as energy consumption.**

***Index Terms*—Data access control, Privacy, Signcryption, re-encryption, security.**

I. Introduction

Implantable medical devices (IMD) are deployed surgically in the patient's body which monitors physiological conditions within them. These devices include implantable cardiac defibrillators (ICDs), pacemakers, nero-simulators, and drug delivery systems, etc. It also can accomplish a variety of ailments, such as cardiac arrhythmia, diabetes, and Parkinson's disease. IMDs also support telemetry transmission for remote monitoring over long-range, high-bandwidth wireless links, and emerging devices will communicate with other inter-operating IMDs. Hence, data security and privacy protection are the main issues that affect medical safety and treatment efficiency. Now as IMDs technology evolve, it is very important to improve security and effectiveness. Over the last decade, the use of wireless sensor networks (WSNs) in healthcare has gained popularity. In addition to reducing costs, WSNs also enable physicians to provide services in remote areas. A major use of WSNs in healthcare is implanting IMDs in the patient's body. These IMDs continuously monitor vitals and administer appropriate treatments in some cases, especially for those elderly people who live independently and face difficulties when it comes to self-care. Nowadays, implantable devices are equipped with radio transceivers to facilitate communication and data information, such as pacemakers. Such interfaces enable medical professionals to get access to the data whenever they need it. However, this has various security and privacy challenges, including access to patient's personal data and the unauthorized modification of IMD parameters [1] [2] [3]. In order to thwart unauthorized access to IMDs, the conventional solutions using public-key cryptosystems or on pre-loaded secret keys cannot be applied directly, since they typically prevent the ambulance staff from accessing the IMD in an emergency [4]. Therefore, IMDs built a tightness between access control, i.e. privacy and safety of the patient.

Numerous authors have proposed several solutions to address this problem. Within a closed range communication, access control solutions have the advantage of simplicity and intuitiveness. But do not ensure any firm guarantees about the higher range of communication. For example, an attacker can use a high-power transmitter and antenna.

A. Motivation and contribution

Proxy re-encryption (PRE) is one of the most essential cryptographic primitives which allows the third party to modify or alter a ciphertext encrypted for a user, such that another user who is participating in the protocol can decrypt it. This mechanism is very useful in several security applications with respect to data storage and data access [5] [6]. PRE preserves confidentiality but does not ensure the other security requirements such as authentication, non-repudiation, and integrity which are also very essential for IMDs security. In 2017, Li et al. [7] proposed a signcryption scheme with PRE based on IBSC. In this work, the data owner delegates to a designated proxy who is allowed to alter a ciphertext generated with the data owner's public key into different ciphertexts which is authorized user can decrypt. Conflicting to the conventional PRE, the authorized user is only allowed to perform the decryption and verification. The scheme is very costly to computation and communication overheads. In addition, since this scheme is based on identity, it suffers from key management and storage issues.

Certificateless cryptosystem can solve the key escrow problem. Based on the certificateless system, the storage space and computational cost can be greatly reduced. In IMDs, the interactive components communicate through the wireless link, and the sensitive patient data is transmitted through un-trusted public channels. The information might be eavesdropped on, altered, tampered or deleted. Therefore, it is essential to preserve the confidentiality, integrity, authentication, and non-repudiation of the security goal's data. The developed scheme should provide the function of public key encryption and digital signature. Therefore, considering the susceptibility of patients' data, it is essential to develop lightweight cryptographic primitives for data access control that ensures all kinds of security goals. Our contributions are as follows.

1) We propose a novel and low-cost signcryption scheme in a certificateless setting with proxy re-encryption and prove the security under the random oracle model. The security analysis shows that the proposed scheme ensures confidentiality, integrity, authentication, and non-repudiation simultaneously.
2) We design a secure data access control scheme for IMDs. The proxy re-encryption technique is used to ensure that patient data is accessed in a secure manner. Since our scheme is based on the certificateless scheme, it eliminates the key escrow which substantially reduces the cost in storage and key management.
3) We choose the machine with the configuration of core i7-5500U 2.4 2.4 GHz processor and memory 16GB and implemented the scheme. Furthermore, empirical analysis is carried out by evaluating computational cost, communication overhead, and energy consumption.

B. Organization

The organization of the paper is as follows. Section II describes the previous related work. Section III discusses the network and its operations. Section IV-A defines the generic model of certificateless signcryption with proxy re-encryption scheme. Section IV-B introduces the security and adversary models. Section V, we design the protocol. Section VI discusses the empirical analysis of the proposed scheme (DAC-PRE). Finally, we briefly describe the conclusion in Section VII.

II. Related Work

Implantable medical devices are framed in two ways. The first model is designed where the medical devices are fully implanted in the human body such as pacemakers and ICDs. The second type joints the medical devices that are partially implanted such as insulin pump systems. The implanted components are sensor devices and simulators. The externals of the patient's body are installed with a battery and processing unit. For secure communications, the current research focuses on the wireless interfaces to communicate IMDs, which exposes the devices to various attacks. These models are classified on their objectives, goals and impact patients' health data and medical devices. These include eavesdropping attack where an adversary can eavesdrop on the messages communicating between the programmer and the IMDs as well as between the IMD components such as MICS transceivers and sensor devices. Since the transmitting messages are not encrypted, the adversary can extract and analyze the data such as information associated with IMD identification, patient's physiological data, history of healthcare-related information and treatment, therapy information, etc. Furthermore, the adversary can also acquire sensitive information, including the credential used fr authentication such as IMD's identification numbers. Another most important attack is unauthorized access to the IMD components by capturing the IMD's PIN. The adversary successfully modifies/alters the various important data such as therapy setting, patient's health data, etc. After successful access, the adversary may interrupt/stop the injection of insulin, leading to a high glucose level in the patient's body. Further, the adversary can also run unauthorized command and modify IMD firmware, alter the data stored in IMD or disable it. These attacks can be mounted by unauthorized IMD programmers or the use of a Universal Software Radio Peripheral. IMD's security requires a right balance between confidentiality, integrity, and availability as per the CIA triad. Further, strong access control mechanisms are required which would provide a trade-off between security and safety a challenge. Hence, it is essential to design efficient access control techniques for IMDs. Many authors proposed many access control schemes. Rasmussen *et* al. developed an access control scheme based on ultrasonic distance-bounding [8]. The scheme allows an IMD to provide secure access to an IMD programmer (reader) within a proximity range. In this scheme, the IMD programmer does not suffer any computational power. However, this does not ensure non-traceability property, session key security, and is also not resistant against replay and man-the-middle attacks. Ellouze et al. [9] proposed secure cardiac IMDs using wireless identification and sensing platform. For IMD's battery life, they performed the harvesting of energy using radio frequency signals from the UHF RFID reader for the generation of the key and authentication. Subsequently, they developed an authentication scheme by generating the bio-metric cryptographic key. This provides secure communication between the IMD programmer and the sensing platform of IMD. However, this scheme does not achieve the security properties anonymity and non-traceability, also not secure against reply attack. Jang *et* al. [10] designed a secure scheme using both symmetric and asymmetric cryptosystems. This provides hybrid security which ensures the different levels of security in wireless body networks. However, this scheme also does not provide security property anonymity.

A. Signcryption with certificateless setting

Signcryption is a cryptographic primitive that performs both the function of digital signature and public-key encryption in one single logical step. It has lower computational cost and communication overhead than the traditional "sign and encrypt" method, vice versa. This is first introduced by Zheng [11]. Malone-Lee developed the first identity-based signcryption with a security framework for constraint devices [12]. However, this scheme suffers from key escrow problem. Subsequently, this issue of key management

and storage is eliminated by [7] [13]. Further Libert and Quisquater indicated some vulnerability in the scheme and then designed a new scheme in [14]. Barbos and Farshim designed the first signcryption scheme with certificatelss setting [15]. Subsequently many authors have proposed certificateless signcryption scheme (CLSC) for various security applications [16] [17].

B. Proxy re-encryption for data access control

There is numerous of application of PRE. This includes a parallel distribution system, e-mail forwarding broadcast encryption with the key establishment for wireless sensor networks, digital rights management, etc. Many authors [18][19] proposed signcryption with PRE having identity-based setting. However, these schemes are a burden of key management and suffer the key-escrow problem. Furthermore, these schemes are mathematically not correct[20]. Chandrasekhar *e*t al. proven that these schemes are not secure against chosen-ciphertext attack. Subsequently, Fagen Li *e*t al.[7] designed an access control for cloud computing that works through IBSC with PRE. Many access control schemes have been proposed which also works via PRE techniques for cloud computing [6][21]. Many constructions are also based on either attribute-based encryption (ABE) or the combination of both ABE and PRE [22].

III. Network Architecture and Components

In this section, we discuss, how the proposed protocol **DAC-PRE** can work in the network model 1. The model consists of the following communicating devices and entities:

- `Key Generator Center`: This is a trusted third party or registration center, where all the entities namely data owners and data users register with their identities. Then KGC constructs the partial private keys for both the users. Since our protocol is based on the certificatelss setting, the full private keys are generated by using a combination function on the secret values and the received partial private keys. Hence it does not suffer key escrow problems and the certificate management burden including storage and visualization. We assume that the KGC is always a trusted entity and can never be compromised in any situation.
- `Implantable Medical device`: The implantable device (IMD) is designed with sensors that can collect various psychological values such as heart rate, blood pressure, temperature. An IMD equipped with a controller is implanted in the patient.
- `Data owner`: The data owner is equipped with a controller. The patient's data is captured by the IMD and is transmitted to the controller. The controller is in the domain of the data owner.
- `Medical gateway device` (MGD): The medical gateway device (MGD) is a high- power server. It is very conscious and trusted support to improve the reliability of the implantable medical devices. It operates data storage services, visualization, and proxy re-encryption. It stores the signcrypted data and has interfaces with the data user. We assume that the server is honest and curious.
- `Data user`: All entities and services related to medical health are act as data users. This includes the doctor, the patient's relatives, or any emergency service like ambulance etc. In real-life medical and healthcare applications, always these entities require the data of the patient for better diagnosis and treatment. For example, the doctor needs the history of the patient, medicine is given, etc. In addition, sanatoriums or hospitals need to check the details of patients when they are admitted to the hospital, and what kind of medical services are provided.

The model operates via the following steps:

1) When a user might be the patient, doctors, other medical services who would like to access the data from the medical gateway device (MGD), it sends a query message to the data owner. He/she gives the instruction to signcrypt the data acquired from IMD. The MGD has high computing power, including the ability to store and process the signature and transmit the signed data to the user. The MGD is considered honest yet curious.
2) The MGD is in the domain of the patients regarded as the owner of the data. It establishes secure communication with the patient's IMD either online or offline using Transmission Layer Security (TLS) approach. It is equipped with a database that stores encrypted data.
3) On the received request, the data owner then issues a delegation command to the MGD for data re-encryption signcrypted data which is stored in MGD , and forwards it to the data user. We assume that the data owner is restricted from any attempts to attach or delete the data collected from the IMD.
4) The data user runs the decryption algorithm to obtain the signcrypted data and transmits it to the cloud server through a secure channel following an offline or online approach using TLS protocol. The cloud server performs the decryption algorithm to verify and allow the requester that is the data user.

A. Security requirements

The proposed protocol **DAC-PRE** is deployed in the model which ensures privacy in the data and authentication. Additionally, it preserves integrity in the data and non-repudiation.

- Data Privacy: Data privacy should be ensured when the data is stored as confidential from unauthorized data users.
- Integrity is preserved, when the data is transmitted from the data owner (the patient) to the data user doctor and other medical services remains unchanged for illegitimate users.
- Authentication: This security goal is achieved when the data remains valid can be verified by the data owner.
- Non-repudiation: This security goal can be ensured when the patient is undeniably linked to past activities with the signcryption of the data.

IV. A Generic Framework of Signcryption with Proxy Re-encryption

This section describes a formal design of the signcryption scheme with proxy re-encryption and discusses the adversary model works in this scheme.

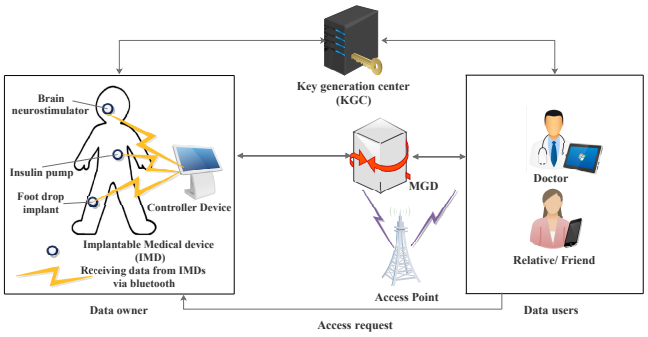

Fig. 1. Network model of IMDs

A. Framework

A certificateless signcryption with proxy re-encryption scheme consists of the following nine algorithms.

- `Setup`: The KGC takes the secret parameter ψ as input and runs this algorithm. This algorithm generates the master secret key s and sets the master public key P_{pub}. Finally, the KGC produces the global system parameters $param$ and keeps s secret.
- `PPr-Ext`: This algorithm works for partial private key generation. The KGC takes the user's identity ID_u and master secret key s as input. This produces the partial private key D_u the corresponding user.
- `PubK-Gen`: This algorithm is used to generate a public key and run it by user. The user first chooses a secret value x_u. The algorithm takes x_u as input and constructs his public key pk_u.
- `PrK-Gen`: This algorithm works to generate the full private key of the user. The algorithm takes the partial private key D_u as input, then chose secret value x_u and constructs the full private key.
- `SC`: This signcryption algorithm generates the signcryt on a given message and works in a designed protocol. The protocol involves two communicating entities Alice and Bob regarded as sender and receiver respectively, along with another entity say Carlo who is allowed to re-encrypt the generated ciphertext. This algorithm takes Alice's identity ID_A, private key S_A, public key pk_A, Bob's identity ID_B, public key pk_B along with the message as input and produces a ciphertext σ_{AB}.
- `PK-Gen`: This algorithm is used to generate a proxy key which is run by the receiver Bob and establishes the proxy key K_{BC}. The algorithm takes the global system parameters $param$, private key S_B together with Carlo's identity ID_C and public key pk_C as inputs.
- `RE-Enc`: This algorithm is used to re-encrypt ciphertexts and run by a proxy taking the first level ciphertext σ_{AB} and proxy key K_{BC} as input. This generates the second level ciphertext σ_{AC} which is valid for Alice and Carlo.
- `USC`: This is a deterministic algorithm run by the receiver Bob to recover the message. This takes $\sigma_{AB}, ID_A, pk_A, pk_B$ and S_B as input and returns the symbol $\perp$ if σ_{AB} is an invalid ciphertext.
- `DEC`: This algorithm is used for the decryption of the second-level encrypted message. This takes σ_{AB}, ID_A, ID_B long with the proxy user Carlo's private key S_C as input and returns m or the symbol $\perp$ for errors when σ_{AC} is an invalid ciphertext.

Formally, we can summarize these algorithms as follows: if $\sigma_{AB} = \texttt{SC}(m, S_A, ID_A, pk_A, pk_B, ID_B)$, then $m = \texttt{USC}(\sigma_{AB}, ID_A, S_B, pk_B, ID_B)$. Additionally, if $K_{BC} = \texttt{PK-Gen}(S_B, pk_C, ID_C)$ and $\sigma_{AC} = \texttt{RE-Enc}(\sigma_{AB}, K_{BC})$, then $m = \texttt{Dec}(\sigma_{AC}, ID_A, ID_B, S_C)$. For sake of simplicity, we have omitted the system parameter $param$.

B. Adversary Model

A certificateless signcryption with proxy re-encryption scheme must achieve the two security goals, confidentiality, and unforgeability. Confidentiality is defined by against chosen ciphertext attack (IND-iCCA-I/II) and unforgeability against adaptive chosen message attack (EUF-CMA). The notion of strong existential unforgeability is denoted by sUF-iCMA-I/II for insider security.

The following subsection IV-C and IV-D define the confidentiality and unforgeability model respectively. We consider the Type-I and Type-II adversaries defined in [15]. A Type-I adversary is not allowed to access the master key. However, he can replace the user's public key with a value chosen by him. A Type-II adversary does not allow to replace the user's public key with any value. However, he can be allowed to access the master secret key. The adversary model is designed on the gameplay between the adversary and challenger. All hash functions are modeled as random oracles. The challenger uses the adversary as a subroutine to attempt to compute a solution of the intractable mathematical hard problem.

C. Model for Confidentiality

This model is described to ensure IND-iCCA-I/II security. Let $\mathscr{A}_I$ denotes Type-I adversary, whereas $\mathscr{A}_{II}$ denotes for Type-II adversary. Furthermore, the interaction between the challenger $\mathscr{C}$ and forger $\mathscr{B}$ with $\mathscr{A}_I$ is modeled by `Game-I`, whereas the interaction with $\mathscr{A}_{II}$ is as `Game-II`.
`Game-I`: This game is played between $\mathscr{A}_I$ and the challenger $\mathscr{C}$. The following steps and phases are to be performed.

- `Setup`: The challenger $\mathscr{C}$ takes the security parameter ψ and executes the setup algorithm, returns the system global parameter $param$, and provides to $\mathscr{A}_I$.
- `Phase-I`: $\mathscr{A}_I$ interacts with $\mathscr{C}$ with submitting polynomially bounded queries in an adaptive manner.
  - **Partial private key queries**: $\mathscr{A}_I$ submits queries with ID. $\mathscr{C}$ runs the `PPr-Ext`, obtains D_{ID} and sends to $\mathscr{A}_I$.
  - **Private key queries**: $\mathscr{A}_I$ submits queries with identity ID. $\mathscr{C}$ runs `Pr-Gen` algorithm, returns the full private key S_{ID} and sends it to $\mathscr{A}_I$.
  - **Public key queries**: $\mathscr{A}_I$ submits queries with identity ID. $\mathscr{C}$ runs `PubK-Gen` algorithm, returns the public key pk_{ID} and sends it to $\mathscr{A}_I$.
  - **Public key replacement queries**: $\mathscr{A}_I$ intends to replace the value of pk_{ID} with a suitable key.

– **Proxy key construction queries**: $\mathscr{A}_I$ chooses two distinct identifies ID_i and ID_j. $\mathscr{C}$ runs PK–Gen and PrK–Gen algorithms, obtains deligator's public key pk_j and private key S_i respectively. Then $\mathscr{C}$ runs PubK–Gen algorithms with these input and obtains the encryption key K_{ij}. Formally PK–Gen$(S_i, pk_j, ID_j) = K_{ij}$. Finally $\mathscr{C}$ sends it to $\mathscr{A}_I$.
– **Signcryption queries**: In this oracle, $\mathscr{A}_I$ submits the queries with tuples (m, ID_i, ID_j). $\mathscr{C}$ generates the sender's private and public key pairs (S_i, pk_i) by running the PrK–Gen and PubK–Gen algorithms respectively. $\mathscr{C}$ calls SC$(m, S_i, pk_i, ID_i, ID_j)$, generates σ_{ij} and sends it to $\mathscr{A}_I$. Note that the oracle accepts $\mathscr{A}_I$ to reveal any secret value, if the user's public key is replaced with a preferred value.
– **Re-encryption queries**: In this oracle, $\mathscr{A}_I$ submits the queries with σ_{ij} along with three identities ID_i, ID_j and ID_u as input. $\mathscr{C}$ first runs PrK–Gen and PubK–Gen algorithms and obtains the private key S_j and the user's public key pk_u. Then $\mathscr{C}$ obtains the proxy key K_{ju} by running the algorithm PK–Gen. Formally, we can write as PK–Gen$(S_j, pk_u, ID_u) = K_{ju}$.
– **Unsigncryption queries**: $\mathscr{A}_I$ submits the queries with tuples $(\sigma_{ij}, ID_i, ID_j)$. $\mathscr{C}$ first runs PrK–Gen and PubK–Gen algorithms and obtains the receiver's private and public key pairs (S_j, pk_j). Then $\mathscr{C}$ calls USC$(\sigma_{ij}, S_j, ID_i, pk_j, ID_j)$, obtains the message m or the symbol $\perp$ for invalid σ_{ij} and sends it to $\mathscr{A}_I$.
– **Decryption queries**: In this oracle, $\mathscr{A}_I$ submits queries with tuples $(\sigma_{iu}, ID_i, ID_j, ID_u)$. $\mathscr{C}$ first obtains the user's private key S_u by running PrK–Gen algorithm. The $\mathscr{C}$ calls DEC$(\sigma_{iu}, S_u, ID_i, ID_j)$ and obtains m and the symbol $\perp$ for invalid ciphertext σ_{iu}.

- Challenge: Here, $\mathscr{A}_I$ aborts the simulation knowingly. For which it selects the challenged identities ID_A and ID_B. Additionally, it chooses two distinct messages m_0 and m_1 of equal lengths with the following conditions.
  1) $\mathscr{A}_I$ does not submit any query with ID_B for private key oracle.
  2) $\mathscr{A}_I$ does not submit query with ID_B for partial private key and key-replacement oracle.
  3) $\mathscr{A}_I$ does not submit query with the tuples (ID_B, ID_u) proxy key and ID_u for private key oracle.
- Phase–II: Similar to the interaction performed in Phase–I between $\mathscr{C}$ and $\mathscr{A}_I$ via queries. $\mathscr{A}_I$ runs the polynomially bounded queries adaptively. But the following constraints are to be applied.
  1) $\mathscr{A}_I$ is not allowed to submit private key query with ID_B.
  2) $\mathscr{A}_I$ is not allowed to submit the partial private key and public key replacement query on ID_B.
  3) $\mathscr{A}_I$ is not allowed to submit the proxy-key generation queries with (ID_B, ID_u) and private key queries on ID_u.
  4) $\mathscr{A}_I$ is not allowed to submit the query to Unsigncryption oracle with tuple $(\sigma_{AB}, ID_A, ID_B)$ when the sender's public key pk_A is replaced after the Challenge phase.
  5) $\mathscr{A}_I$ is not allowed to submit the queries to re-encryption oracle with tuple $(\sigma_{AB}, ID_A, ID_B, ID_u)$ and private key oracles with ID_u.
  6) $\mathscr{A}_I$ is not allowed to submit the queries to re-encryption oracle with tuple $(\sigma_{AB}, ID_A, ID_B, ID)u)$. Where the response is σ_{Au}. Additionally $\mathscr{A}_I$ is not allowed to submit private key oracles with $(\sigma_{Au}, ID_A, ID_B, ID_u)$.
- Guess: $\mathscr{A}_I$ obtains a bit $\beta^{'}$ and wins the game when $\beta^{'} = \beta$. The $\mathscr{A}_I$'s advantages is defined as

$$\mathrm{Adv}^{\mathrm{IND-iCCA-I}}_{\mathrm{DAC-PRE}}(\mathcal{A}_I) = |2\Pr[\beta' = \beta] - 1|,$$

where $\Pr[\beta' = \beta]$ denotes the probability that $\beta' = \beta$.

Game–II: $\mathscr{A}_{II}$ plays this game with $\mathscr{C}$ via the polynomially bounded queries in an adaptive manner similar to the Game–I.

– Setup: The challenger $\mathscr{C}$ takes security parameter ψ as input, executes the Setup algorithm and obtains the system global parameter $param$. Then sends it to $\mathscr{A}_I$.
– Phase–I: $\mathscr{A}_{II}$ interacts with $\mathscr{C}$ with submitting polynomially bounded queries in an adaptive manner. Note that here $\mathscr{A}_{II}$ does not submit partial private key queries since $\mathscr{A}_{II}$ possesses the master secret key s and can generated partial private key.
– Challenge: Here, $\mathscr{A}_{II}$ aborts the simulation knowingly. For which it selects the challenged identities ID_A and ID_B. Additionally, it chooses two distinct messages m_0 and m_1 of equal lengths with the following conditions.
  1) $\mathscr{A}_{II}$ does not submit any query with ID_B for private key oracle.
  2) $\mathscr{A}_{II}$ does not submit query with (ID_B, ID_u) for proxy key and does not submits queries with ID_u private key oracle.
  3) $\mathscr{A}_{II}$ does not submit query with the tuples (ID_B, ID_u) proxy key and ID_u for private key oracle.
– Phase–II: Similar to the interaction performed in Phase–I between $\mathscr{C}$ and $\mathscr{A}_{II}$ via queries. $\mathscr{A}_{II}$ runs the polynomially bounded queries in an adaptive manner with the following constraints:
  1) $\mathscr{A}_{II}$ is not allowed to submit private key query with ID_B.
  2) $\mathscr{A}_{II}$ is not allowed to submit the proxy-key generation queries with (ID_B, ID_u) and private key queries on ID_u.
  3) $\mathscr{A}_{II}$ is not allowed to submit the query to Unsigncryption oracle with tuple $(\sigma_{AB}, ID_A, ID_B)$

when the sender's public key pk_A is replaced after the Challenge phase.

4) $\mathscr{A}_{II}$ is not allowed to submit the queries to re-encryption oracle with tuple $(\sigma_{AB}, ID_A, ID_B, ID_u)$ and private key oracles with ID_u.
5) $\mathscr{A}_{II}$ is not allowed to submit the queries to re-encryption oracle with tuple $(\sigma_{AB}, ID_A, ID_B, ID)u)$. Where the response is σ_{Au}. Additionally $\mathscr{A}_{II}$ is not allowed to submit decryption oracle with $(\sigma_{Au}, ID_A, ID_B, ID_u)$ for identity ID_u.

– Guess: $\mathscr{A}_{II}$ obtains a bit $\beta^{'}$ and wins the game when $\beta^{'} = \beta$. The $\mathscr{A}_{II}$'s advantages is defined as

$$\mathrm{Adv}_{\mathrm{DAC-PRE}}^{\mathrm{IND-iCCA-II}}(\mathcal{A}_{II}) = |2\Pr[\beta' = \beta] - 1|,$$

where $\Pr[\beta' = \beta]$ denotes the probability that $\beta' = \beta$.

Definition 1. An certificateless signcryption with proxy re-encryption achieves IND-iCCA-I/II security with respect to Game−I and Game−II, if there exists the polynimially bounded adversaries $\mathscr{A}_I$ and $\mathscr{A}_{II}$ have negligible advantage in these games.

D. Model for Unforgeability

In this model, the security is sUF-iCMA-I/II. Similarly, here we also describe two games Game−I played between $\mathscr{F}_I$ and $\mathscr{C}$ and Game−II, played between $\mathscr{F}_{II}$ and $\mathscr{C}$.

- Setup: The challenger $\mathscr{C}$ takes the security parameter ψ as input, executes the setup and returns the system global parameter $param$. Then send it to $\mathscr{F}_I$.
- Attack: $\mathscr{F}_I$ interacts with $\mathscr{C}$ with submitting polynomially bounded queries in an adaptive manner.
- Forgery: Here, $\mathscr{F}_I$ obtains a tuples $(ID_A, ID_B, \sigma_{AB})$ and wins the game with the following conditions.
  1) m = USC$(\sigma_{AB}, ID_A, S_B, ID_B)$.
  2) $\mathscr{F}_I$ does not submit query with ID_A to private key oracle.
  3) $\mathscr{F}_I$ does not submit query with (ID_A, ID_u) for proxy key and also does not submits queries with ID_u to private key oracle.
  4) $\mathscr{F}_I$ does not submit signcryption queries with (m, ID_A, ID_u) that generates the ciphertext σ_{Au}. Note that the decryption of the corresponding ciphertext is a valid one, where both ID_u and ID_B are distinct.

The $\mathscr{F}_I$'s advantage is defined by the probability of success. Now we discuss the Game−II played between $\mathscr{F}_{II}$ and $\mathscr{C}$.

– Setup: The challenger $\mathscr{C}$ takes the security parameter ψ as input, executes the setup algorithm and obtains the system global parameter $param$. Then sends it to $\mathscr{F}_{II}$.
– Attack: $\mathscr{F}_{II}$ interacts with $\mathscr{C}$ with submitting polynomially bounded queries in an adaptive manner.
– Forgery: Here, $\mathscr{F}_{II}$ obtains a tuples $(ID_A, ID_B, \sigma_{AB})$ and wins the game with the following conditions.
  1) m = USC$(\sigma_{AB}, ID_A, S_B, ID_B)$.
  2) $\mathscr{F}_{II}$ does not submit query with ID_A to private key oracle.
  3) $\mathscr{F}_{II}$ does not submit query with (ID_A, ID_u) for proxy key and also does not submits queries with ID_u to private key oracle.
  4) $\mathscr{F}_{II}$ does not submit signcryption query with (m, ID_A, ID_u) that generates the ciphertext σ_{Au}. Note that the decryption of the corresponding ciphertext is a valid one, where both ID_u and ID_B are distinct.

The $\mathscr{F}_{II}$'s advantage is the probability of success.

Definition 2. A certificateless signcryption with proxy re-encryption achieves sUF-iCMA-I/II security with respect to the unforgeability Game−I and Game−II, if there exists the polynimially bounded adversaries $\mathscr{F}_I$ and $\mathscr{F}_{II}$ have negligible advantage in these games.

V. Construction of Protocol

The protocol are being executed in three different phases as **Protocol Initialization**, **Protocol messages** and **Protocol action**. All global system parameters are set by KGC in the protocol initialization and it is one-time setup. The proposed scheme comprises the following six algorithms namely **Signcryption**, **Proxy-key construction**, **Re-encryption**, **Unsigncryption** and **Decryption**. These are executed via three phases that are described in subsection V-A, V and V-B.

A. Protocol Initialization

This is the system initialization phase which is performed by Setup algorithm taking a security parameter ψ, where $\psi \geq 160$ bits. The KGC sets two groups as G_1 and G_2 of prime order $q \geq 2^{\psi}$, where the group G_1 is an additive group with generator P and G_2 is a multiplicative group. The KGC selects a bilinear map $\hat{e} : G_1 \times G_1 \rightarrow G_2$. This also chooses collision resistant hash functions $H_1 : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$, $H_2 : \{0,1\}^* \times G_2 \times G_1 \rightarrow \mathbb{Z}_q^*$, $H_3 : G_2 \rightarrow \mathbb{Z}_q^*$, $H_4 : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$, where n denotes the size of the message on which the algorithm works to generate the signcrypt. The KGC constructs the master public key $P_{pub} = s \cdot P$, where $s \in \mathbb{Z}_q^*$ is chosen at random as the master secret key. The KGC publishes $\{G_1, G_2, P_{pub}, n, H_1, H_2, H_3, H_4\}$ as the system parameters and keeps master secret key s as secret. The protocol works through the KGC. The user participating in this protocol has to use his identity to register with KGC. The KGC uses the master secret key and computes the partial private key D_u of the corresponding user which is given by $D_u = sQ_u$, where $Q_u = H_1(ID_u)$ and u denotes to any arbitrary user. Then D_u is transmitted to the corresponding user via secure channel using TLS protocol. After received the partial private key, the user randomly selects $x_u \in \mathbb{Z}_q^*$ and constructs full private key as $S_u = (x_u, D_u)$. The user's public key is $pk_u = x_u P$.

B. Protocol action

The following four algorithms are run in this phase by three users. Let the users which participate in this protocol are `Alice`, `Bob` and `Carlo`. Let the identity of these users are denoted by ID_A, ID_B and ID_C respectively.

1) `Signcryption`: The algorithm is executed by `Alice`. It takes the identity ID_A, private key S_A of Alice, the public key, identity ID_B of Bob, and the public system parameter U as inputs, then return the ciphertext. The algorithm details are as follows.

Algorithm 1 Signcrypt

: INPUT: (m, S_A, pk_B, U, ID_B)
: OUTPUT: $\sigma_{AB} \leftarrow (C_1, C_2, Z)$
begin
  Chooses $r \in \mathbb{Z}_q^*$ at random.
  Compute $U \leftarrow r \cdot P$.
  Compute $T \leftarrow \hat{e}(P_{pub}, Q_B)^r$.
  Compute $T_1 \leftarrow rpk_B$.
  Set $h \leftarrow H_2(U, T, T_1, ID_B, pk_A, pk_B)$
  Compute $C_1 \leftarrow m \oplus h$
  Set $W_1 \leftarrow H_3(U, C_1, T_1, ID_A, pk_A)$.
  Set $W_2 \leftarrow H_4(U, C_1, T_1, ID_A, pk_A)$.
  Set $W \leftarrow W_1 \oplus W_2$.
  Set $k \leftarrow H_5(U, T)$.
  Compute $C_2 \leftarrow W \oplus k$.
  Compute $Z \leftarrow D_A + rW_1 + x_A W_2$
  Return : $\sigma_{AB} \leftarrow (C_1, C_2, Z)$
end

2) `Proxy-key construction` : The protocol allows both Bob and Carlo to generate proxy-key k_{BC}. The algorithm takes Bob's private key $S_B = (x_B, D_B)$, additionally Carlo's public key pk_C as input and construct k_{BC}. The algorithm proceeds as Algorithm 2.

Algorithm 2 Proxy-key Construction

: INPUT: $S_B = (x_B, D_B)$, U, pk_C
: OUTPUT: k_{BC}
begin
  Compute $T_1 \leftarrow x_B U$.
  Compute $T \rightarrow \hat{e}(D_B, U)$.
  Compute $W_1 \leftarrow H_3(U, C_1, T_1, ID_A, pk_A)$.
  Compute $W_2 \leftarrow H_4(U, C_1, T_1, ID_A, pk_A)$.
  Choose $r_1 \in \mathbb{Z}_q^*$ and compute $T_2 \leftarrow r_1 pk_C$.
  Compute $R \leftarrow r_1 P$.
  Compute $T^{'} \leftarrow \hat{e}(P_{pub}, Q_C)^{r_1}$.
  Set $\bar{W} \leftarrow H_5(T^{'}, U, R, T_2, ID_B, pk_C)$.
  Compute $W \leftarrow W_1 \oplus W_2$.
  Compute $X = \bar{W} \oplus W$.
  Compute $k_{BC} \leftarrow (T_2, X)$.
end

3) `Re-encryption` : The algorithm takes σ_{AB} and k_{BC} as input and re-encrypts the ciphertext σ_{AC} . The algorithm proceeds as

Algorithm 3 Re-encryption

: INPUT: $\sigma_{AB} \leftarrow (C_1, C_2, Z)$
: OUTPUT: $\sigma_{AC} \leftarrow (C_1, C_1^{'}, C_2, Z)$
begin
  Compute $C_1^{'} \leftarrow C_1 + T_2$.
  Compute $C_2^{'} \leftarrow C_2 \oplus X$.
  Return : $\sigma_{AC} \leftarrow (C_1, C_1^{'}, C_2, C_2^{'}, Z)$.
end

4) `Unsigncryption` : The algorithm is run by Bob with his own private and public key and additionally Alice's public key. The key k used for decryption is verified and is used in the deciphering process. The algorithm produces the ciphertext and signature for the given message. This proceeds as

Algorithm 4 Unsigncrypt

: INPUT: $\sigma_{AB} \leftarrow (C_1, C_2, Z)$
: OUTPUT: $(0/1)$
begin
  Compute $T_1 \leftarrow x_B U$.
  Compute $T = \hat{e}(D_B, U)$.
  Compute $h \leftarrow H_2(U, T, T_1, ID_B, pk_A, pk_B)$
  Compute $W_1 \leftarrow H_3(U, C_1, T_1, ID_A, pk_A)$.
  Compute $W_2 \leftarrow H_4(U, C_1, T_1, ID_B, pk_B)$.
  Set $W \leftarrow W_1 \oplus W_2$.
  Compute $k \leftarrow H_5(U, T)$.
  if $W = C_2 \oplus k$ **then**
    Return "1"
  else
    Return "0"
  end if
  Compute $m \leftarrow C_1 \oplus h$
  if $\hat{e}(P, Z) = \hat{e}(P_{pub}, Q_A)\hat{e}(U, W_1)\hat{e}(pk_A, W_2)$ **then**.
    Return "1" and Accept m.
  else
    Return "0" and Reject.
  end if
end

5) `Decryption`: This algorithm takes σ_{AC} and Carlo's private and public key. Additionally this takes Bob's public as input. This decipher obtains the message and verifies the signature.

Theorem 1. *In the random oracle model, the proposed* **DAC-PRE** *protocol is IND-iCCA-I/ II secure under the assumption that the GBDH problem is intractable in the underline bilinear group.*

This is followed from Lemma -1 and Lemma -2

Lemma 1. *In random oracle model, suppose there is an probabilistic polynomial time adversary of Type-I denoted by $\mathscr{A}_I$ has advantage ϵ win the IND-iCCA-I game against the proposed protocol* **DAC-PRE** *at a time t while submitting the q_s signcryption queries, q_{pk} proxy key queries, q_u unsigncryption queries, q_e re-encryption queries, q_d decryption queries and hash queries q_{H_i}, for $i = 1, 2, 3, 4, 5$ where the hash functions are modelled as random oracle, then there exists an*

Algorithm 5 Decryption

: INPUT: $\{\sigma_{AC} = (C_1, C_1^{'}, C_2, C_2^{'}, Z)\}$, k_{BC}
: OUTPUT: $(0/1)$.
begin
Compute $T_2 \leftarrow x_C R$.
Compute $T^{'} \leftarrow \hat{e}(D_C, R)$.
Compute $h \leftarrow H_2(U, T, T_1, ID_B, pk_A, pk_B)$
Compute $W_1 \leftarrow H_3(U, C_1, T_1, ID_B, pk_B)$.
Compute $W_2 \leftarrow H_4(U, C_1, T_1, ID_A, pk_A)$.
$\bar{W} \leftarrow H_5(T^{'}, U, R, T_2, ID_B, pk_C)$.
Compute $C_1 \leftarrow C_1^{'} - T_2$.
Compute $m \leftarrow C_1 \oplus h$.
Compute $k \leftarrow C_2^{'} \oplus \bar{W}$.
if $\hat{e}(P, Z) = \hat{e}(P_{pub}, Q_A)\hat{e}(U, W_1)\hat{e}(pk_A, W_2)$ **then**
Return "1" and Accept m.
else
Return "0" and Reject.
end if
end

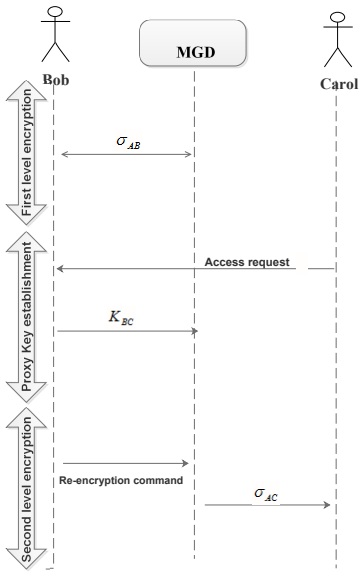

Fig. 2. Data access control in IMD

algorithm $\mathscr{B}$ which uses $\mathscr{A}_I$ to solve GBDH problem with advantage

$$\epsilon_{gbdh} \geq \left(\frac{\epsilon}{q_{H_1}}\right)\left(1 - \frac{q_s(q_{H_2}+q_{H_3}+q_{H_4}+q_{H_5})}{2^{\psi}}\right)\left(1 - \frac{q_u}{2^{\psi}}\right)$$

in a time $t^{'} \leq t + \mathcal{O}(q_{pk} + q_s + q_e + q_u + q_d)t_p$, where t_p denotes the computational cost for pairing operation.

Proof. See Appendix 1 □

Lemma 2. *In random oracle model, suppose there is an probabilistic polynomial time adversary of type-II denoted by $\mathscr{A}_{II}$ has advantage ϵ win the IND-iCCA-II game against the proposed protocol* **DAC-PRE** *at a time t, while submitting q_s signcryption queries, proxy key construction queries q_{pk}, q_u unsigncryption queries, q_e re-encryption queries, q_d decryption queries and hash queries q_{H_i}, for $i = 1, 2, 3, 4$ in which the hash functions are modelled as random oracle, then there exists an algorithm $\mathscr{B}$ which uses $\mathscr{A}_{II}$ to solve CDH problem with advantage*

$$\epsilon_{cdh} \geq \left(\frac{\epsilon}{q_{H_1}^2}\right)\left(1 - \frac{q_s(q_{H_2}+q_{H_3}+q_{H_4}+q_{H_5})}{2^{\psi}}\right)\left(1 - \frac{q_u}{2^{\psi}}\right)$$

in a time $t^{'} \leq t + \mathcal{O}(q_{pk} + q_s + q_e + q_u + q_d)t_p$, where t_p denotes the cost of one pairing operation.

In above equation, $q_\delta = q_{H_1} + q_{H_2} + 2q_{ppk} + q_s + q_u$ denotes the maximum number of queries the adversary could submit to H_1, H_2, partial private key oracle, signcryption oracle and unsigncryption oracle.

Proof. See Appendix 2 □

Theorem 2. *In the random oracle model, the proposed* **DAC-PRE** *protocol is sUF-iCMA-I/ II secure under the assumption that the CDH problem is intractable in the additive group G_1.*

This is followed from Lemma -3 and Lemma -4

Lemma 3. *Under random oracle model, suppose there is an probabilistic polynomial time adversary of Type-I denoted by $\mathscr{F}_I$ has advantage ϵ wins the sUF-iCMA-I game against the proposed protocol* **DAC-PRE** *at a time t submitting q_s signcryption queries, q_{pk} proxy key generation queries, q_u unsigncryption queries, q_{re} re-encryption queries, q_d decryption queries and hash queries q_{H_i}, for $i = 1, 2, 3, 4, 5$ where the hash functions are modelled as random oracle, then there exists an algorithm $\mathscr{B}$ which uses $\mathscr{F}_I$ to solve CDH problem with advantage*

$$\epsilon_{cdh} \geq \frac{10(q_s+1)(q_s+q_{H_3}+q_{H_4})q_{H_1}}{2^{\psi}-1}$$

in a time $t^{'} \leq \Psi q_{H_1} q_{H_3} q_{H_4}(t + \mathcal{O}(q_{pk} + q_s + q_e + q_u + q_d)t_p/\epsilon(1 - \frac{1}{2^{\psi}})$
where $\Psi = 120686$ and t_p denotes the pairing cost.

Proof. See Appendix 3 □

Lemma 4. *In random oracle model, suppose there is an probabilistic polynomial time adversary of type-II having probabilistic polynomial time denoted by $\mathscr{F}_{II}$ has advantage ϵ wins the sUF-iCMA-II game against the proposed protocol* **DAC-PRE** *at a time t submitting q_s signcryption queries, q_{pk} proxy key generation queries, q_u unsigncryption queries, q_{re} re-encryption queries, q_d decryption queries and hash queries q_{H_i}, for $i = 1, 2, 3, 4, 5$ where the hash functions are modelled as random oracle, then there exists an algorithm $\mathscr{B}$ which uses $\mathscr{A}_I$ to solve CDH problem with advantage*

$$\epsilon_{cdh} \geq \left(\frac{\epsilon}{q_{H_1}^2}\right)\left(1 - \frac{q_s(q_{H_2}+q_{H_5})}{2^{\psi}}\right)\left(1 - \frac{q_u}{2^{\psi}}\right)$$

in a time $t^{'} \leq t + \mathcal{O}(q_{pk} + q_s + q_e + q_u + q_d)t_p$, where t_p denotes the cost of pairing operation.

Proof. See Appendix 4 □

VI. Performance Evaluation

We evaluate the computational and communication overhead of our DAC-PRE scheme and assess against the certificateless scheme SBTS [19] and CLSPRE [23] as indicated in Table III and Table V respectively. For empirical analysis, we evaluate the computational cost of all these three schemes.

The implementation is done using PBC library [24]. We have chosen Type A following elliptic curve with embedding degree is two. The additive group G_1 of order p of size 160 bit with generator P consists of the elements with points lies on the elliptic curve defined over 512-bit prime finite field.

$$y^2 \equiv (x^3 + x) \bmod q$$

where $q \equiv 3 \bmod 4$. Let we denote the operation of ECC point multiplication in the additive group G_1 as M, pairing as P, hashing as H and modular exponentiation as ME. Table III represents the computational cost evaluated in term of the cryptographic operations. The following machine with configurations has been consider in our implementation and done simulation.

- Processor- core i7-5500U 2.4 2.4 GHz processor.
- Memory- RAM of 16GB.

In order to evaluate the execution time, each algorithm is run 4000 times and the average is taken. From table IV, we can observe that the our scheme DAC-PRE is $(64.13 - 62.50)/64.13 = 2.54\%$ faster than SBTS [19], $(79.11 - 62.50)/79.11 = 20.99\%$ faster than CLSPRE [23] and $(74.52 - 62.50)/74.52 = 16.12\%$ faster than CLS-PRE [25]. Therefore, our scheme DAC-PRE is more efficient with respect to the execution time than the other certificateless schemes on access control designed on the random oracle model. The communication overhead is evaluated for σ_{AB} and σ_{AC} at 80-bit, 112-bit and 128-bit key size security levels. According to these security level specification, the size of the two prime p and q are given in Table I [26]. Similarly the size of the group elements, integers and message sizes are specified in Table II at 80-bit, 112-bit and 128-bit key size security levels. The communication overhead for σ_{AB} and σ_{AC} are based on Table I and Table II. Let the size of the arbitrary element $Q \in G_1$ is denoted by $|Q|$. Similarly $|u|$ and $|x|$ denote the size of element belonging to group G_2 and $\mathbb{Z}_q^*$ respectively. Assume that the size of the message $|\ m\ | = 160$ bits. When we measure the security strength of 80-bit security, the size of p, q, G_1 and G_2 is 160, 512, 1024 and 1024 bits respectively. The size of the group element in G_1 can be reduced to size 65 bytes by using the compression technique [27]. Therefore the communication overhead at 80-bit security level of SBTS [19] is $3 * |G_1| = 3 * 65 = 195$ bytes for σ_{AB}. Similarly the communication overhead of CLSPRE [23], CLS-PRE [25] and our scheme DAC-PRE are $2 * |G_1| + 2 * |m| = 2 * 65 + 2 * 20 = 170$ bytes, $2*|G_1|+|\mathbb{Z}_q^*|+|m| = 2*65+2*20+2*20 = 170$ bytes and $|G_1|+2*|m| = 2*65+20 = 150$ bytes for σ_{AB} respectively. The communication overhead of SBTS [19], CLSPRE [23], CLS-PRE [25] and DAC-PRE for $sigma_{AC}$ are $3 * |G_1| = 3*65 = 195$ bytes, $3*|G_1|+2*|m| = 3*65+2*20 = 170$ bytes, $3*|G_1|+|\mathbb{Z}_q^*|+|m| = 3*65+20+20 = 235$ bytes and $|G_1|+4*|m| = 65+4*20 = 145$ bytes for σ_{AC} respectively at 80-bit security level. Similarly the communication overhead is evaluated at 112-bit and 128-bit security level. Fig 4 and Fig 5 summarize the communication overhead of SBTS [19], CLSPRE [23], CLS-PRE [25] and DAC-PRE for σ_{AB} and σ_{AC} respectively at 80-bit, 112-bit and 128-bit security level.

TABLE I
SECURITY LEVEL SPECIFICATION IN BITS

Security level	Size of p	Size of q

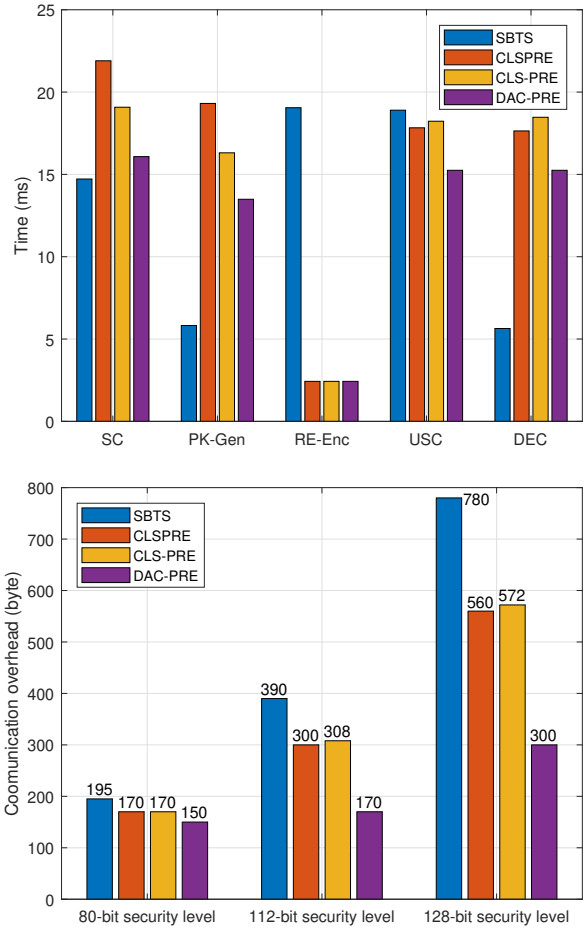

Fig. 4. The communication overhead for σ_{AB}

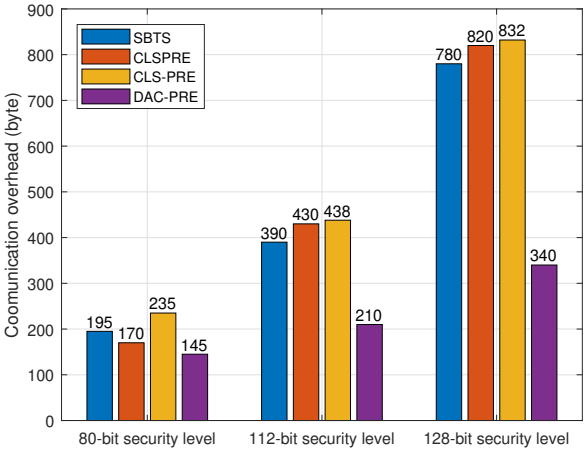

Fig. 5. The communication overhead for σ_{AC}

TABLE III
COMPUTATION COST

Scheme	SC	PK-Gen	RE-Enc	USC	DEC
SBTS [19]	3M	1M	5M	4M	1M
CLSPRE [23]	1P+3M	2P+3M	N/A	1P+2M	1P+2M
CLS-PRE [25]	4M	6M	N/A	4M	4M
DAC-PRE	1P+2M	2P+2M	N/A	1P+1M	1P+1M

TABLE IV
COMPUTATIONAL EXECUTION TIME IN SECOND

Scheme	SC	PK-Gen	RE-Enc	USC	DEC
SBTS [19]	14.72	5.82	19.05	18.9	5.64
CLSPRE [23]	21.9	19.31	2.43	17.83	17.64
CLS-PRE [25]	19.08	16.31	2.43	18.23	18.47
DAC-PRE	16.08	13.49	2.43	15.25	15.25

TABLE V
COMMUNICATION OVERHEAD

Scheme	Communication overhead		Formal security	
	σ_{AB}	σ_{AC}	IND-CCA2	EUF-CMA
SBTS [19]	$3\|G_1\|$	$3\|G_1\|$	$\times$	$\surd$
CLSPRE [23]	$2\|G_1\| + 2\|m\|$	$3\|G_1\| + 2\|m\|$	$\surd$	$\surd$
CLS-PRE [25]	$2\|G_1\| + \|\mathbb{Z}_q^*\| + \|m\|$	$3\|G_1\| + \|\mathbb{Z}_q^*\| + \|m\|$	$\surd$	$\surd$
DAC-PRE	$\|G_1\| + 2\|m\|$	$\|G_1\| + 4\|m\|$	$\surd$	$\surd$

VII. CONCLUSION

We have designed an efficient and low-cost data access control scheme for IMDs. The proposed scheme is based on the certificateless signcryption with proxy re-encryption technique. The security is proven under the random oracle model. The scheme provides IND-CCA2 and EUF-CMA security under the assumption of GBDH and CDH problems. Furthermore, our scheme ensures the security goals of confidentiality, authentication, non-repudiation, and integrity. From the experimental analysis, we have shown that the proposed scheme is more efficient to computational cost than the other relevant schemes.

VIII. ACKNOWLEDGMENT

This work is supported by Anusandhan National Research Foundation (ANRF), DST Govt of India under the scheme of State University Research Excellence with Project ID-SUR/2022/004771 and under National Foreign Specialized Program of the Ministry of Science and Technology of China with Project ID-DL2022186001L.

REFERENCES

[1] R. Spring, E. Freudenthal, and L. Estevez, "Practical techniques for limiting disclosure of rf-equipped medical devices," in *2007 IEEE Dallas Engineering in Medicine and Biology Workshop*. IEEE, 2007, pp. 82–85.

[2] D. Halperin, T. S. Heydt-Benjamin, K. Fu, T. Kohno, and W. H. Maisel, "Security and privacy for implantable medical devices," *IEEE pervasive computing*, vol. 7, no. 1, pp. 30–39, 2008.

[3] D. Halperin, T. S. Heydt-Benjamin, B. Ransford, S. S. Clark, B. Defend, W. Morgan, K. Fu, T. Kohno, and W. H. Maisel, "Pacemakers and implantable cardiac defibrillators: Software radio attacks and zero-power defenses," in *2008 IEEE Symposium on Security and Privacy (sp 2008)*. IEEE, 2008, pp. 129–142.

[4] S. K. Gupta, T. Mukherjee, and K. Venkatasubramanian, "Criticality aware access control model for pervasive applications," in *Fourth Annual IEEE International Conference on Pervasive Computing and Communications (PERCOM'06)*. IEEE, 2006, pp. 5–pp.

[5] L. Jiang, L. Da Xu, H. Cai, Z. Jiang, F. Bu, and B. Xu, "An iot-oriented data storage framework in cloud computing platform," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 1443–1451, 2014.

[6] A. N. Khan, M. M. Kiah, S. A. Madani, M. Ali, S. Shamshirband *et al.*, "Incremental proxy re-encryption scheme for mobile cloud computing environment," *The Journal of Supercomputing*, vol. 68, no. 2, pp. 624–651, 2014.

[7] F. Li, B. Liu, and J. Hong, "An efficient signcryption for data access control in cloud computing," *Computing*, vol. 99, no. 5, pp. 465–479, 2017.

[8] K. B. Rasmussen, C. Castelluccia, T. S. Heydt-Benjamin, and S. Capkun, "Proximity-based access control for implantable medical devices," in *Proceedings of the 16th ACM conference on Computer and communications security*, 2009, pp. 410–419.

[9] N. Ellouze, M. Allouche, H. Ben Ahmed, S. Rekhis, and N. Boudriga, "Securing implantable cardiac medical devices: Use of radio frequency energy harvesting," in *Proceedings of the 3rd international workshop on Trustworthy embedded devices*, 2013, pp. 35–42.

[10] C. S. Jang, D. G. Lee, J.-w. Han, and J. H. Park, "Hybrid security protocol for wireless body area networks," *Wireless Communications and Mobile Computing*, vol. 11, no. 2, pp. 277–288, 2011.

[11] Y. Zheng, "Digital signcryption or how to achieve cost (signature & encryption) $\ll$ cost (signature)+ cost (encryption)," in *Annual international cryptology conference*. Springer, 1997, pp. 165–179.

[12] J. Malone-Lee, "Identity-based signcryption." *IACR Cryptol. ePrint Arch.*, vol. 2002, p. 98, 2002.

[13] C. Wang and X. Cao, "An improved signcryption with proxy re-encryption and its application," in *2011 Seventh International Conference on Computational Intelligence and Security*. IEEE, 2011, pp. 886–890.

[14] C. Wu and Z. Chen, "A new efficient certificateless signcryption scheme," in *2008 International Symposium on Information Science and Engineering*, vol. 1. IEEE, 2008, pp. 661–664.

[15] M. Barbosa and P. Farshim, "Certificateless signcryption," in *Proceedings of the 2008 ACM symposium on Information, computer and communications security*, 2008, pp. 369–372.

[16] Z. Cheng and R. Comley, "Efficient certificateless public key encryption." *IACR Cryptol. ePrint Arch.*, vol. 2005, p. 12, 2005.

[17] W. Xie and Z. Zhang, "Efficient and provably secure certificateless signcryption from bilinear maps," in *2010 IEEE International Conference on Wireless Communications, Networking and Information Security*. IEEE, 2010, pp. 558–562.

[18] V. Kirtane and C. P. Rangan, "Rsa-tbos signcryption with proxy re-encryption," in *Proceedings of the 8th ACM workshop on Digital rights management*, 2008, pp. 59–66.

[19] P. Shabisha, A. Braeken, A. Touhafi, and K. Steenhaut, "Elliptic curve qu-vanstone based signcryption schemes with proxy re-encryption for secure cloud data storage," in *International Conference of Cloud Computing Technologies and Applications*. Springer, 2017, pp. 1–18.

[20] W. Huige, W. Caifen, and C. Hao, "Id-based proxy re-signcryption scheme," in *2011 IEEE International Conference on Computer Science and Automation Engineering*, vol. 2. IEEE, 2011, pp. 317–321.

[21] X. Tian, X. Wang, and A. Zhou, "Dsp re-encryption: A flexible mechanism for access control enforcement management in daas," in *2009 IEEE International Conference on Cloud Computing*. IEEE, 2009, pp. 25–32.

[22] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving secure, scalable, and fine-grained data access control in cloud computing," in *2010 Proceedings IEEE INFOCOM*. Ieee, 2010, pp. 1–9.

[23] E. Ahene, Z. Qin, A. K. Adusei, and F. Li, "Efficient signcryption with proxy re-encryption and its application in smart grid," *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 9722–9737, 2019.

[24] B. Lynn, "Pbc library manual 0.5. 11," 2006.

[25] E. Ahene, J. Dai, H. Feng, and F. Li, "A certificateless signcryption with proxy re-encryption for practical access control in cloud-based reliable

smart grid," *Telecommunication Systems*, vol. 70, no. 4, pp. 491–510, 2019.
[26] J. Kar, K. Naik, and T. Abdelkader, "An efficient and lightweight deniably authenticated encryption scheme for e-mail security," *IEEE Access*, vol. 7, pp. 184 207–184 220, 2019.
[27] N.-W. Lo and J.-L. Tsai, "An efficient conditional privacy-preserving authentication scheme for vehicular sensor networks without pairings," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 5, pp. 1319–1328, 2015.
[28] D. Pointcheval and J. Stern, "Security arguments for digital signatures and blind signatures," *Journal of cryptology*, vol. 13, no. 3, pp. 361–396, 2000.



Jayaprakash Kar has received his M.Sc and M.Phil in Mathematics from Sambalpur University, M.Tech and Ph.D in Computer Science (Cryptographic Protocols) from Utkal University, India. He is visiting researcher of Department of Electrical & Computer Engineering, University of Waterloo, Canada. Currently he is working as Associate Professor in the Department of Computer Science & Engineering, The LNM Institute of Information Technology, Jaipur, India. He is Center-Lead of "Center for Cryptology, Cybersecurity and Digital forensics" (C3-SDF). His current research interests are on Cryptographic protocols and primitives using Elliptic Curve and Pairing based Cryptography in Random Oracle and Standard model, Zero-knowledge proofs, secret sharing, and multi-party computation. Dr. Kar is advisory and editorial board member of many peer reviewed Journals and international conferences. He is Associate editor of Journal of Circuits, Systems and Computers, World Scientific. He is life member of International Association for Cryptology Research (IACR), Cryptology Research Society of India, IEEE senior member, and associate member of ACM, International Association of Computer Science & Information Technology (Singapore) and International Association of Engineers (United Sates).



Xiaoguang Liu is now a lecturer in the School of Mathematics, Southwest Minzu University, Chengdu, P.R. China. He received his Ph.D. degree in mathematics from Shaanxi normal university, Xi'an, P.R. China in 2014. From 2017 to 2020, he is a postdoctoral fellow in the School of Computer Science and Engineering, University of Electronic Science and Technology of China (UESTC), Chengdu, P.R. China. His recent research interests include cryptography, network security, and optimization computation.



Fagen Li (Member, IEEE) received the Ph.D. degree in cryptography from Xidian University, Xi'an, China, in 2007. He is currently a Professor with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China. From 2008 to 2009, he was a Postdoctoral Fellow with Future University-Hakodate, Hokkaido, Japan, which is supported by the Japan Society for the Promotion of Science. From 2010 to 2012, he was a Research Fellow with the Institute of Mathematics for Industry, Kyushu University, Fukuoka, Japan. He has authored or coauthored more than 110 papers in international journals and conferences. His research interests include cryptography and network security.

APPENDIX

Proof of Lemma: 1

Proof. The algorithm $\mathscr{B}$ uses $\mathscr{A}_I$ as subroutine to solve GBDH problem on the given instances (P, aP, bP, cP). $\mathscr{B}$ interacts with $\mathscr{A}_I$ in the following ways.

`Setup`: The `Setup` in the initialization phase is run by $\mathscr{B}$ with the security parameter ψ taking as input, obtains the $param$. The algorithm $\mathscr{B}$ sets $P_{pub} = aP$ and $params = (\psi, P_{pub})$, then transmits them to $\mathscr{A}_I$.The hash functions H_i, $i = 1, 2, 3, 4, 5$ are modelled as random oracle. For consistency in the answer of hash queries, $\mathscr{B}$ maintains the lists L_i $i = 1, 2, 3, 4, 5$ to store the answer that are being submitted to the respective oracles.
`Phase-I`: $\mathscr{A}_I$ submits polynomially bounded queries to all oracles and $\mathscr{B}$ answers all queries in the following ways:

- H_1 Queries: Algorithm $\mathscr{B}$ selects an index l uniformly at random in $\{1, 2 \dots q_{H_1}\}$. Assume that $\mathscr{A}_I$ submits queries to H_1-oracle on distinct identities ID_i, where i sets to counter 1. $\mathscr{A}_1$ does not submit the query on an identity in , if it has been queried and answers the stored value in L_1 for that particular identity. $\mathscr{B}$ picks ID_l at random from $\{1, 2 \dots q_{H_1}\}$ as the challenged identity. $\mathscr{B}$ answers $\mathscr{A}_I$ as
  1) At the lth query, $\mathscr{B}$ answers with $H_1(ID_l) = bP$ and adds the tuple $(ID_l, bP, \perp)$ to L_1.
  2) For other queries submitted on ID_i to H_1 oracle *i*.e for $H_1(ID_i)$ queries, where $i \neq l \in \{1, 2 \dots q_{H_1}\}$. $\mathscr{B}$ chooses $b_i \in \mathbb{Z}_q^*$ at random, adds the tuples (ID_i, b_iP, b_i) to L_1 and answers $H_1(ID_i) = b_iP$ to $\mathscr{A}_I$.
- H_2 Queries: $\mathscr{A}_I$ submits the queries with $(U, T, T_1, ID_B, pk_A, pk_B)$ to H_2 oracle. Algorithm $\mathscr{B}$ proceeds as follows:
  1) It verifies if the DBDH oracle returns 1, when the query is being submitted with tuple (aP, bP, cP, T), if it true then $\mathscr{B}$ returns T and abort.
  2) Algorithm $\mathscr{B}$ searches through the list L_2 with entries $(U, *, T_1, ID_B, pk_B, W_1)$ for distinct values of W_1, such that the DBDH oracle returns 1 when the query is submitted with tuple (aP, bP, cP, T). In this case, note that $ID_B = ID_l$. If such a tuple exists, then it returns W_1 and replaces the symbol $*$ with T.
  3) If the algorithm $\mathscr{B}$ founds this stage, it generates a W_1 at random and updates the list L_2, which is initially empty with the respective tuple that compromises the input and returns values.
- H_3 Queries: When $\mathscr{A}_I$ submits queries with $(U, C_1, T_1, ID_A, pk_A)$ to H_3 oracle, algorithm $\mathscr{B}$ first searches the tuple in L_3, $\mathscr{B}$ returns the answer if there exists such a tuple in L_3. Otherwise,$\mathscr{B}$ selects a random value t, return tP, and updates the list L_3.
- H_4 Queries: $\mathscr{A}_I$ submits queries with $(U, C_1, T_1, ID_A, pk_A)$ to H_4 oracle, algorithm $\mathscr{B}$ first searches the tuple in L_4, if founds return the answer, otherwise picks a random value w, returns wP and update the list L_4.
- H_5 Queries: $\mathscr{A}_I$ submits queries with (U, T). Algorithm $\mathscr{B}$ first searches the tuple in L_5, if founds, then reply k to $\mathscr{A}_I$. Else $\mathscr{B}$ chooses $k \in \{0, 1\}^n$ at random, outputs k and update L_5 with (U, T, k).
- `Partial Private Key Queries`: $\mathscr{A}_I$ submits queries with ID_i, algorithm $\mathscr{B}$ call H_1 with ID_i and proceeds as
  1) If $ID_i = ID_l$, $\mathscr{B}$ aborts the simulation.
  2) If $ID_i \neq ID_l$, then $\mathscr{B}$ computes $D_i = b_iaP$ and adds the tuple $(ID_i, \perp, D_i, \perp)$ to L_{pk}. Where L_k is used for private key oracle that store all the answers and the identity ID_i on which the queries is submitted.
- `Public Key Queries`: $\mathscr{A}_I$ submits queries with ID_i. Algorithm $\mathscr{B}$ searches the entry (ID_i, D_i, pk_i, x_i) in L_{ppk}, if found then $\mathscr{B}$ sends pk_i to $\mathscr{A}_I$. Else $\mathscr{B}$ generates a new tuple by choosing $x_i \in \mathbb{Z}_q^*$ at random, compute $pk_i = x_iP$, adds $(ID_i, \perp, pk_i, x_i)$ to L_k and sends to $\mathscr{A}_I$.
- `Public Key Replacement Queries`: $\mathscr{A}_I$ submits queries with ID_i for replacement of a valid public key, algorithm $\mathscr{B}$ updates L_{pk} with $(ID_i, \perp, pk_i, \perp)$. Algorithm $\mathscr{B}$ can use this value for subsequent computations.
- `Private Key Queries`: $\mathscr{A}_I$ submits queries with ID_i . Assume that there will no replacement of a valid public key queries is submitted with ID_i and $ID_i \neq ID_l$. Algorithm $\mathscr{B}$ searches the entry (ID_i, pk_i, x_i) in L_{pk} and answers with $S_i = (x_i, b_iaP)$ to $\mathscr{A}_I$. If the query for public key replacement is not submitted with ID_i, then $\mathscr{A}_I$ will not be allowed to access the private key S_i. If $ID_i = ID_l$, then algorithm $\mathscr{B}$ aborts the simulation and stop.
- `Proxy Key Construction Queries`: $\mathscr{A}_I$ submits queries with ID_i and ID_j. Algorithm $\mathscr{B}$ computes the delegater's private key S_i. It chooses randomly $\alpha, \beta \in \mathbb{Z}_q^*$ and the element W_2 from G_1 at random. Then sets $Z = \beta P_{pub} + x_iW_2, W_1 = \alpha P_{pub}$ and $U = (\beta P - Q_i)\alpha^{-1}$. Computes $R = r_1P$, $T^{'} = \hat{e}(D_j, R)$ and $T_2 = r_1pk_j$. $\mathscr{C}$ retrieves $\bar{W}$ from L_5 list and computes $X = \bar{W} \oplus W$. Finally this will return the proxy key $k_{ij} = (T_2, X$ as answer. Consequently $\mathscr{B}$ fails to return the proxy key when $i = l$.
- `Signcryption Queries`: Let the sender and receiver's identities are denoted by ID_A and ID_B respectively. Let the m denote the message on which the cipher is to be generated. $\mathscr{A}_I$ submits queries with the tuple (m, ID_A, ID_B). The algorithm $\mathscr{B}$ executes and returns an answer in the following ways. There are two cases in this simulation.
  1) If $ID_A \neq ID_l$, algorithm $\mathscr{B}$ can obtain the private key S_A and computes σ_{AB}. Then $\mathscr{B}$ answers σ_{AB} to $\mathscr{A}_I$.
  2) If $ID_A = ID_l$, algorithm $\mathscr{B}$ chooses $r \in \mathbb{Z}_q^*$ at random and computes $U = rP, T = \hat{e}(D_B, U)$ and $T_1 = rpk_B$. Selects the two elements

W_1 and W_2 at random from G_1 and computes $W = W_1 \oplus W_2$. Then computes $Z = bP_{pub} + x_A W_2 + rW_1$. Sets $k = H_4(U, T)$ and defines $h = H_2(U, T, T_1, ID_B, pk_A, pk_B)$. Finally computes $C_1 = m \oplus h$ and $C_2 = W \oplus k$. The algorithm returns $\sigma_{AB} = (C_1, C_2, Z)$ to $\mathscr{A}_I$.

- `Re-encryption Queries`: $\mathscr{A}_I$ submits the queries with $(\sigma_{AB}, ID_A, ID_B, ID_C)$. Algorithm $\mathscr{B}$ proceeds on the following two cases:
  1) If $ID_B \neq ID_l$, It executes re-encryption and generates σ_{AC} through the queries submitted with (ID_B, ID_C).
  2) If $ID_B = ID_l$, $\mathscr{B}$ chooses the element $T_2 \in G_1$, define $W_1 = H_3(U, C_1, T_1, ID_B, pk_B)$ and $W_2 = H_4(U, C_1, T_1, ID_A, pk_A)$. Then computes $W = W_1 \oplus W_2$ and define $\bar{W} = H_5(T^{'}, U, R, T_2, ID_B, pk_C)$. Finally computes $X = \bar{W} \oplus W$, compute $C_1^{'} = C_1 + T_2$ and $C_2^{'} = C_2 \oplus X$. Finally, the algorithm $\mathscr{B}$ returns $\sigma_{AC} = (C_1, C_1^{'}, C_2, Z)$ to $\mathscr{A}_I$.
- `Unsigncryption Queries`: $\mathscr{A}_I$ submits the queries on $\sigma_{AB} = (C_1, C_1^{'}, C_2, Z)$ with ID_A and ID_B. Algorithm $\mathscr{B}$ proceeds as:
  1) If $ID_B \neq ID_l$, then it obtains S_B by calling private key oracle with ID_B, completes Unsigncryption and returns m to $\mathscr{A}_I$.
  2) When $ID_B = ID_l$, then the pairing cannot be computed. In order to follow consistency in the answer, $\mathscr{B}$ searches the entry $(U, T, T_1, ID_B, pk_B, h)$ for distinct value of T such that the decision bilinear Diffie-Hellman oracle returns 1 when the query submitted with (aP, bP, cP, U, T). If such an entry exists, then the corresponding value of U and T can be found. Algorithm $\mathscr{B}$ calls H_4 and obtains k. The checks whether $W = C_2 \oplus k$ holds. If it false then $\mathscr{B}$ returns "0" and aborts the simulations. Otherwise, algorithm $\mathscr{B}$ calls H_2, obtains h, computes $C_1 \oplus h$ and obtains m. Finally to accept m, algorithm $\mathscr{B}$ checks if $\hat{e}(P, Z) = \hat{e}(P_{pub}, Q_A)\hat{e}(U, W_1)\hat{e}(pk_A, W_2)$ holds and returns m to $\mathscr{A}_I$, otherwise it returns "0" for invalid ciphertext.
  3) If $\mathscr{B}$ searches this point of execution, it chooses a random h on the list L_2 and places the entry $(U, *, T_1, ID_B, pk_B)$ and decrypts using this h as $C_1 \oplus h$. Note that the symbol $*$ denotes the unknown value of pairing. Finally verifies if $\hat{e}(P, Z) = \hat{e}(P_{pub}, Q_A)\hat{e}(U, W_1)\hat{e}(pk_A, W_2)$ holds and returns m to $\mathscr{A}_I$. Otherwise it returns "0" for invalid ciphertext.
- `Decryption Queries`: $\mathscr{A}_I$ submits the queries on $\sigma_{AC} = (C_1, C_1^{'}, C_2, Z)$ with ID_A, ID_B and ID_C. Algorithm $\mathscr{B}$ proceeds as:
  1) If $ID_C \neq ID_l$, it requests private key oracles and obtains S_C. Then computes m and returns to $\mathscr{A}_I$.
  2) If $ID_C = ID_l$, then $\mathscr{B}$ goes to L_5 and looks for the tuple $(T_\xi^{'}, U, R_\xi, T_{2\xi}, ID_B, pk_C, \bar{W}_\xi)$ indexed by $\xi \in \{1, 2 \ldots q_{H_5}\}$. The algorithm $\mathscr{B}$ takes T_2 and computes $C_1 = C_1^{'} - T_2$ for $T_2 = T_{2\xi}$. If $T_2 \neq T_{2\xi}$ then the algorithm $\mathscr{B}$ looks for next element of L_5, else computes $m = C_1 \oplus h$ and $k = C_2^{'} \oplus \bar{W}_\xi$). The algorithm $\mathscr{B}$ defines $h = H_2(U, T, T_1, ID_B, pk_A, pk_B)$ and checks if $\hat{e}(P, Z) = \hat{e}(P_{pub}, Q_A)\hat{e}(U, W_1)\hat{e}(pk_A, W_2)$ holds. Then returns m to $\mathscr{A}_I$. Else $\mathscr{B}$ looks next element of L_5, if it does not found such message m in the complete searching then it returns "0" and terminate the simulation.
- `Challenge`: Eventually, $\mathscr{A}_I$ returns two messages (m_0, m_1) of equal length. and two identities ID_A^* and ID_B^*. Algorithm $\mathscr{B}$ looks the list L_1 with input ID_B^*. If $ID_B^* \neq ID_l$, then $\mathscr{B}$ fails and aborts the simulation. Otherwise it proceeds to generate a challenge as follows:
  1) $\mathscr{B}$ looks the public key pk_A^* corresponding to ID_A^* in L_{pk}. Then it sets $U^* = cP$, chooses the elements $T_1^*, W_1^*, W_2^* \in G_1$ at random and $k^* \in \{0,1\}^n$. Then chooses hash value h^* and set $W^* = W_1^* \oplus W_2^*$.
  2) Picks a random bit $\varphi \in \{0, 1\}$ and set $C_1^* = m_\varphi \oplus h^*$, $C_2^* = W^* \oplus k^*$.
  3) The component Z^* is set to $D_A + rW_1 + x_A W_2 = D_A + \omega cP + \tau pk_A^*$. Where $\mathscr{B}$ takes ω and τ from L_3 and L_4 respectively. D_A is computed by calling partial private key extraction oracle on ID_A^*. Note that, since $ID_A^* \neq ID_B^*$, the partial private key extraction oracles returns correct value of D_A. Finally $\mathscr{B}$ returns $\sigma_{AB}^* = (C_1^*, C_2^*, Z^*)$ to $\mathscr{A}_I$.
- `Phase-II`: As in Phase-I, $\mathscr{A}_I$ submits a polynomial number of queries adaptively and the algorithm $\mathscr{B}$ answers as a response to all queries to $\mathscr{A}_I$.
- `Guess`: Eventually, $\mathscr{A}_I$ returns its guess as to which message is signcrypted during the challenge. The adversary returns an identity ID_B^* with index l, since l is independent of adversary's view with probability $\frac{1}{q_{H_1}}$. If this event occurs, the simulation would be success unless it submits the queries with challenged-related tuples $(U^*, T^*, T_1^*, ID_B^*, pk_B^*)$ to H_2 oracle. However H_2 is modelled as random oracle, if the tuples does not belong to L_2 list, the adversary does not have any advantage. In this case $\mathscr{B}$ will win the game by solving **GBDH** problem in the simulation of H_2 oracle. The total number of **DBDH** oracle calls, that $\mathscr{B}$ can perform is bounded by at most $q_u^2 + q_u q_{H_2} + q_{H_2}$.

`Probability analysis`: Consider the following independent events that occur in the simulation. Now based on these events, we evaluate $\mathscr{B}$'s success.

- E_1 : $\mathscr{A}_I$ does not take ID_l as the challenged identity.
- E_2 : $\mathscr{A}_I$ submits the queries with ID_l to Partial private key oracle.
- E_3 : $\mathscr{A}_I$ submits the queries with ID_l to Private key oracle.
- E_4 : $\mathscr{A}_I$ submits the queries with (ID_l, ID_j) to Proxy key oracle.

- E_5 : Due to collision resistant properties of all H_i, $i = 2,3,4,5$. $\mathscr{B}$ aborts the simulation in signcryption query.
- E_6 : The algorithm $\mathscr{B}$ aborts the simulation in unsigncryption query because it might reject a valid cipher text. Note that the occurring of events $\neg E_1 \Rightarrow \neg E_2, \neg E_3, \neg E_4$ and $\neg E_5$. So the probability of occurring these events are given by
  - $\Pr[\neg E_1] = \Pr[\neg E_2] = \Pr[\neg E_3] = \Pr[\neg E_4] = \frac{1}{q_{H_1}}$
  - $\Pr[\neg E_5] = 1 - \frac{q_s(q_{H_2}+q_{H_3}+q_{H_4}+q_{H_5})}{2^{\psi}}$
  - $\Pr[\neg E_6] = (1 - \frac{q_u}{2^{\psi}})$.

  Hence $\Pr[\neg E_1 \neg E_2 \neg E_3 \neg E_4 \neg E_5 \neg E_6]$ $= \left(\frac{1}{q_{H_1}}\right)\left(1 - \frac{q_s(q_{H_2}+q_{H_3}+q_{H_4}+q_{H_5})}{2^{\psi}}\right)(1 - \frac{q_u}{2^{\psi}})$. This implies,

  $\epsilon_{gbdh} \geq \left(\frac{\epsilon}{q_{H_1}}\right)\left(1 - \frac{q_s(q_{H_2}+q_{H_3}+q_{H_4}+q_{H_5})}{2^{\psi}}\right)(1 - \frac{q_u}{2^{\psi}})$

  in a time $t^{'} \leq t + \mathcal{O}(q_{pk} + q_s + q_e + q_u + q_d)t_p$

□

Proof of Lemma: 2

Proof. The algorithm $\mathscr{B}$ uses $\mathscr{A}_{II}$ as a subroutine to solve CDH problem on the given instances (P, aP, bP). $\mathscr{B}$ is allowed to access CDH oracle and interacts with $\mathscr{A}_{II}$ in the following ways.

`Setup`: The `Setup` in the initialization phase is run by $\mathscr{B}$ with the security parameter ψ taking as input, to obtain the $param$. The algorithm $\mathscr{B}$ sets $P_{pub} = sP$ and $params = (\psi, P_{pub})$. Then transmit to $\mathscr{A}_{II}$. The hash functions H_i, $i = 1,2,3,4,5$ are modelled as random oracles. For consistency in the answer of hash queries, $\mathscr{B}$ maintains the lists L_i $i = 1,2,3,4,5$ to store the answer that of the queries submitted to the respective oracles. The algorithm $\mathscr{B}$ also maintains one more list L_k for private key oracle.

`Phase-I`: $\mathscr{A}_{II}$ submits polynomially bounded queries in an adaptive manner to all oracles and $\mathscr{B}$ answers these queries in the following ways:

- H_1 Queries: Algorithm $\mathscr{B}$ selects an index l uniformly at random in $\{1,2\ldots q_{H_1}\}$. Assume that $\mathscr{A}_{II}$ submits queries to H_1-oracle on distinct identities ID_i, where i sets to counter 1. $\mathscr{A}_{II}$ does not submit the query on an identity , if it has been queried and answers the stored value in L_1 for that particular identity as described in lemma 1. $\mathscr{A}_{II}$ chooses (ID_l, ID_γ) as challenge identities from the set $\{1,2\ldots q_{H_1}\}$. The algorithm $\mathscr{B}$ chooses $b_i \in \mathbb{Z}_q^*$ at random, and sets $H_1(ID_i) = b_iP$, then add (ID_i, b_iP, b_i) to list L_1 which is initially empty and returns b_iP to $\mathscr{A}_{II}$ as answer to the query submitting with ID_i.
- `Public Key Queries`: $\mathscr{A}_{II}$ submits queries with ID_i. The algorithm $\mathscr{B}$ call private key oracle and searches the entry (ID_i, pk_i, x_i) in L_k, if found then $\mathscr{B}$ sends pk_i to $\mathscr{A}_{II}$. Else $\mathscr{B}$ chooses $x_i \in \mathbb{Z}_q^*$ at random and performs the following
  1) If $ID_i = ID_l$, algorithm $\mathscr{B}$ answers $pk_l = x_iaP$ and if $ID_i = ID_\gamma$, $\mathscr{B}$ responses as $pk_\gamma = x_ibP$.
  2) If $ID_i \neq ID_l$ and If $ID_i \neq ID_\gamma$, algorithm constructs a new pairs (x_i, pk_i), where $pk_i = x_iP$ and add the tuple (ID_i, pk_i, x_i) to L_k.
- H_2 Queries: $\mathscr{A}_{II}$ submits the queries with $(U, T, T_1, ID_B, pk_A, pk_B)$ to H_2 oracle. Algorithm $\mathscr{B}$ proceeds as follows:
  1) It searches the tuples $(U, T, *, ID_B, pk_B, h)$ for distinct value of h, where $\hat{e}(x_iaP, x_ibP) = \hat{e}(x_i^2P, T_1)$. If so, there exists such tuple and replace $*$ with T_1. Note that $ID_i = ID_l$ and $ID_i = ID_\gamma$.
  2) If $\mathscr{B}$ searches this point of execution, It generates a random h and updates the list L_2 with this tuple and the answer value h. Note that initially, the list was empty.
- H_3 Queries: $\mathscr{A}_{II}$ submits queries with $(U, C_1, T_1, ID_A, pk_A)$ to H_3 oracle, algorithm $\mathscr{B}$ first searches the tuple in L_3, if founds return the answer, otherwise picks a random value t, returns tP and update the list L_3.
- H_4 Queries: $\mathscr{A}_{II}$ submits queries with $(U, C_1, T_1, ID_A, pk_A)$ to H_4 oracle, algorithm $\mathscr{B}$ first searches the tuple in L_4, if founds return the answer, otherwise picks a random value w, returns wP and update the list L_4.
- H_5 Queries: $\mathscr{A}_{II}$ submits queries with (U, T), algorithm $\mathscr{B}$ searches the list L_5 for the tuples (U, T, k), if it founds, then return k to $\mathscr{A}_{II}$. Else $\mathscr{B}$ picks $k \in \{0,1\}^n$ at random and replies this to $\mathscr{A}_{II}$ and update L_5 with (U, T, k).
- `Private Key Queries`: $\mathscr{A}_{II}$ submits queries with (x_i, D_i), algorithm $\mathscr{B}$ calls H_1 with ID_i and proceeds as
  1) If $ID_i = ID_l$ or $ID_i = ID_\gamma$, $\mathscr{B}$ aborts the simulation and stop.
  2) If $ID_i \neq ID_l$, ID_γ , $\mathscr{B}$ searches the list L_k and returns $S_i = (x_i, D_i)$ to $\mathscr{A}_{II}$.
- `Proxy Key Construction Queries`: $\mathscr{A}_{II}$ submits queries with ID_i and ID_j. Algorithm $\mathscr{B}$ computes the delegater's private key S_i. It chooses randomly $\alpha, \beta \in \mathbb{Z}_q^*$ and the element W_2 from G_1 at random. Then sets $Z = \beta P_{pub} + x_iW_2, W_1 = \alpha P_{pub}$ and $U = (\beta P - Q_i)\alpha^{-1}$. Computes $R = r_1P, T^{'} = \hat{e}(D_j, R)$ and $T_2 = r_1pk_j$. $\mathscr{C}$ retrieves $\bar{W}$ from L_5 list and computes $X = \bar{W} \oplus W$. Finally this will return the proxy key $k_{ij} = (T_2, X$ as answer. Consequently $\mathscr{C}$ fails to return the proxy key when $i = l$ or $i = \gamma$.
- `Signcryption Queries`: Let the sender and receiver's identities be denoted by ID_A and ID_B respectively. Let the m denote the message on which the cipher is to be generate. $\mathscr{A}_{II}$ submits queries with the tuple (m, ID_A, ID_B). The algorithm $\mathscr{B}$ executes and answers in this ways. It has two cases.
  1) If $ID_A \neq ID_l$ and $ID_A \neq ID_\gamma$, algorithm $\mathscr{B}$ can obtain the private key S_A and computes σ_{AB}. Then $\mathscr{B}$ answers σ_{AB} to $\mathscr{A}_{II}$.
  2) If $ID_A = ID_l$ or $ID_A = ID_\gamma$, algorithm $\mathscr{B}$ chooses $r \in \mathbb{Z}_q^*$ at random and computes $U = rP, T = \hat{e}(D_B, U)$ and $T_1 = rpk_B$. Selects

the two elements W_1 and W_2 at random from G_1 and computes $W = W_1 \oplus W_2$. Then computes $Z = bP_{pub} + x_A W_2 + rW_1$. Sets $k = H_4(U, T)$ and define $h = H_2(U, T, T_1, ID_B, pk_A, pk_B)$. Finally computes $C_1 = m \oplus h$ and $C_2 = W \oplus k$. The algorithm returns $\sigma_{AB} = (C_1, C_2, Z)$ to $\mathscr{A}_{II}$.

- `Re-encryption Queries`: $\mathscr{A}_{II}$ submits the queries with $(\sigma_{AB}, ID_A, ID_B, ID_C)$. Algorithm $\mathscr{B}$ proceeds on the following two cases:
  1) If $ID_B \neq ID_l, ID_\gamma$, It executes re-encryption and generates σ_{AC} through the queries submitted with (ID_B, ID_C).
  2) If $ID_B = ID_l$ or $ID_B = ID_\gamma$ and $ID_C \neq ID_l, ID_\gamma$, $\mathscr{B}$ chooses the element $T_2 \in G_1$, define $W_1 = H_3(U, C_1, T_1, ID_B, pk_B)$ and $W_2 = H_4(U, C_1, T_1, ID_A, pk_A)$. Then computes $W = W_1 \oplus W_2$ and define $\bar{W} = H_5(T^{'}, U, R, T_2, ID_B, pk_C)$. Finally computes $X = \bar{W} \oplus W$, compute $C_1^{'} = C_1 + T_2$ and $C_2^{'} = C_2 \oplus X$. So finally, the algorithm $\mathscr{B}$ returns $\sigma_{AC} = (C_1, C_1^{'}, C_2, Z)$ to $\mathscr{A}_{II}$.
- `Unsigncryption Queries`: $\mathscr{A}_{II}$ submits the queries on $\sigma_{AB} = (C_1, C_1^{'}, C_2, Z)$ with ID_A and ID_B. Algorithm $\mathscr{B}$ proceeds as:
  1) If $ID_B \neq ID_l$, then it obtains S_B by calling private key oracle with ID_B, runs Unsigncryption and returns m to $\mathscr{A}_{II}$.
  2) When $ID_B = ID_l$ or $ID_B = ID_\gamma$, the pairing cannot be computed. In order to follow consistency in all answers, $\mathscr{B}$ searches the entry $(U, T, T_1, ID_B, pk_A, pk_B, h)$ for distinct value of T such that the decision bilinear Diffie-Hellman oracle returns 1 when the query submitted with (aP, bP, cP, U, T). If such an entry exists, the corresponding value of U and T can be found. Algorithm $\mathscr{B}$ calls H_4 and obtains k. The checks whether $W = C_2 \oplus k$ holds. If it fails then $\mathscr{B}$ returns "0" and aborts the simulations. Otherwise, algorithm $\mathscr{B}$ calls H_2, obtains h, computes $C_1 \oplus h$ and recovers m. Finally to accept m, algorithm $\mathscr{B}$ checks if $\hat{e}(P, Z) = \hat{e}(P_{pub}, Q_A)\hat{e}(U, W_1)\hat{e}(pk_A, W_2)$ holds and returns m to $\mathscr{A}_I$, otherwise it returns "0" for invalid ciphertext.
  3) If $\mathscr{B}$ searches this point of execution, it chooses a random h on the list L_2 and puts the entry $(U, *, T_1, ID_B, pk_B)$ and decrypts using this h as $C_1 \oplus h$. Note that the symbol $*$ denotes the un-known value of pairing. Finally it verifies if $\hat{e}(P, Z) = \hat{e}(P_{pub}, Q_A)\hat{e}(U, W_1)\hat{e}(pk_A, W_2)$ holds and returns m to $\mathscr{A}_I$. Otherwise it returns "0" for invalid ciphertext.
- `Decryption Queries`: $\mathscr{A}_{II}$ submits the queries on $\sigma_{AC} = (C_1, C_1^{'}, C_2, Z)$ with ID_A, ID_B and ID_C. Algorithm $\mathscr{B}$ proceeds as:
  1) If $ID_C \neq ID_l$, it requests private key oracles and obtains S_C. Then it computes m and returns to $\mathscr{A}_I$.
  2) If $ID_C = ID_l$, then $\mathscr{B}$ goes to L_5 and looks for the tuple $(T_\xi^{'}, U, R_\xi, T_{2\xi}, ID_B, pk_C, \bar{W}_\xi)$ indexed by $\xi \in \{1, 2 \ldots q_{H_5}\}$. The algorithm $\mathscr{B}$ takes T_2 and computes $C_1 = C_1^{'} - T_2$ for $T_2 = T_{2\xi}$. If $T_2 \neq T_{2\xi}$ then the algorithm $\mathscr{B}$ looks for next element of L_5, else computes $m = C_1 \oplus h$ and $k = C_2^{'} \oplus \bar{W}_\xi$). The algorithm $\mathscr{B}$ defines $h = H_2(U, T, T_1, ID_B, pk_A, pk_B)$ and checks if $\hat{e}(P, Z) = \hat{e}(P_{pub}, Q_A)\hat{e}(U, W_1)\hat{e}(pk_A, W_2)$ holds. Then returns m to $\mathscr{A}_I$. Else $\mathscr{B}$ looks next element of L_5, if it does not found such message m in the complete searching then it returns "0" and terminate the simulation.
- `Challenge`: Eventually, $\mathscr{A}_{II}$ returns two messages (m_0, m_1) of equal length. and two identities ID_A^* and ID_B^*. Algorithm $\mathscr{B}$ looks the list L_1 with input ID_B^*. If $ID_B^* \neq ID_l$, then $\mathscr{B}$ fails and aborts the simulation. Otherwise it proceeds to generate a challenge as follows:
  1) $\mathscr{B}$ looks the public key pk_A^* corresponding to ID_A^* in L_k. Then it sets $U^* = cP$, chooses the elements $T_1^*, W_1^*, W_2^* \in G_1$ at random and $k^* \in \{0,1\}^n$. Then chooses hash value h^* and set $W^* = W_1^* \oplus W_2^*$.
  2) Picks a random bit $\varphi \in \{0, 1\}$ and set $C_1^* = m_\varphi \oplus h^*$, $C_2^* = W^* \oplus k^*$.
  3) The component Z^* is set to $D_A + rW_1 + x_A W_2 = D_A + \omega cP + \tau pk_A^*$, where $\mathscr{B}$ takes ω and τ from L_3 and L_4 respectively. D_A is computed by calling partial private key extraction oracle on ID_A^*. Note that, since $ID_A^* \neq ID_B^*$, the partial private key extraction oracles returns correct value of D_A. Finally $\mathscr{B}$ returns $\sigma_{AB}^* = (C_1^*, C_2^*, Z^*)$ to $\mathscr{A}_{II}$.
- `Phase-II`: As in Phase-I, $\mathscr{A}_{II}$ submits a polynomial number of queries adaptively and the algorithm $\mathscr{B}$ answers as a response to all queries to $\mathscr{A}_I$.
- `Guess`: Eventually, $\mathscr{A}_{II}$ returns its guess as to which message is signcrypted during the challenge. The adversary returns an identity ID_B^* with index l, since l is independent of adversary's view with probability $\frac{1}{q_{H_1}}$. If this event occurs, the simulation would be success unless it submits the queries with challenged-related tuples $(U^*, T^*, T_1^*, ID_B^*, pk_A^*, pk_B^*)$ to H_2 oracle. Where $T_1 = x_i^2 abP, pk_A^* = x_i aP$ and $pk_B^* = x_i bP$. Note that H_2 is modelled as random oracle, if the tuples does not belong to L_2 list, the adversary does not have any advantage. The algorithm $\mathscr{B}$ checks the following equation

$$\hat{e}(x_i^2 P, T_1) = \hat{e}(x_i aP, x_i bP).$$

  If it holds then the algorithm aborts and stops the simulation. This returns the output $T_1^* = abP$. Thus one of them would be the answer to the CDH problem. Thus $\mathscr{B}$ returns wins the IND-iCCA-II game solving **CDH** problem in the simulation of H_2 .

`Probability analysis`: Consider the following independent events that occur in the simulation. Now based on these events, we evaluate $\mathscr{B}$'s successfully.

- E_1 : $\mathscr{A}_I$ does not take ID_l as the challenged identity.

- E_2 : $\mathscr{A}_I$ submits the queries with ID_l to Partial private key oracle.
- E_3 : $\mathscr{A}_I$ submits the queries with ID_l to Private key oracle.
- E_4 : $\mathscr{A}_I$ submits the queries with (ID_l, ID_j) to Proxy key oracle.
- E_5 : Due to collision resistant properties of all H_i, $i = 2, 3, 4, 5$. $\mathscr{B}$ aborts the simulation in signcryption query.
- E_6 : The algorithm $\mathscr{B}$ aborts the simulation in unsigncryption query because it might reject a valid cipher text.

Note that the occurring of events $\neg E_1 \Rightarrow \neg E_2, \neg E_3, \neg E_4$ and $\neg E_5$. So the probability of occurring these events are given by

- Pr[$\neg E_1$] = Pr[$\neg E_2$] = Pr[$\neg E_3$] = Pr[$\neg E_4$] = $\frac{1}{q_{H_1}}$
- Pr[$\neg E_5$] = $1 - \frac{q_s(q_{H_2}+q_{H_3}+q_{H_4}+q_{H_5})}{2^\psi}$
- Pr[$\neg E_6$] = $(1 - \frac{q_u}{2^\psi})$.

Hence Pr[$\neg E_1 \neg E_2 \neg E_3 \neg E_4 \neg E_5 \neg E_6$] = $\left(\frac{1}{q_{H_1}^2}\right)\left(1 - \frac{q_s(q_{H_2}+q_{H_3}+q_{H_4}+q_{H_5})}{2^\psi}\right)(1 - \frac{q_u}{2^\psi})$. This implies,

$\epsilon_{cdh} \geq \left(\frac{\epsilon}{q_{H_1}}\right)\left(1 - \frac{q_s(q_{H_2}+q_{H_3}+q_{H_4}+q_{H_5})}{2^\psi}\right)(1 - \frac{q_u}{2^\psi})$

in a time $t^{'} \leq t + \mathcal{O}(q_{pk} + q_s + q_e + q_u + q_d)t_p$

□

Proof of Lemma: 3

Proof. The algorithm $\mathscr{B}$ uses $\mathscr{F}_I$ which takes the CDH instances (P, aP, bP) as input and computes abP. In the proposed scheme (**DAC-PRE**), to show the authenticity property, the signature is simulated without using the sender's private key and master secret key. The simulation shows how to solve the CDH problem in counterfeiting. The algorithm $\mathscr{B}$ uses $\mathscr{F}_I$ and proceeds as follows.

- `Setup`: Algorithm $\mathscr{B}$ takes the CDH instances (P, aP, bP). $\mathscr{B}$ sets $P_{pub} = bP$, $b \in \mathbb{Z}_q^*$ and provides P_{pub} with system parameters $param$.
- H_1 Queries: Algorithm $\mathscr{B}$ selects an index l uniformly at random in $\{1, 2 \ldots q_{H_1}\}$. Assume that $\mathscr{F}_I$ submits queries to H_1-oracle on distinct identities ID_i. Where i sets to counter 1. $\mathscr{F}_I$ does not submit the query on an identity, if it has been queried and answers the stored value in L_1 for that particular identity as described in lemma 1. $\mathscr{F}_I$ chooses (ID_l, ID_γ) as challenge identities from the set $\{1, 2 \ldots q_{H_1}\}$. The algorithm $\mathscr{B}$ chooses $b_i \in \mathbb{Z}_q^*$ at random, and sets $H_1(ID_i) = b_iP$, then adds (ID_i, b_iP, b_i) to list L_1 which is initially empty and returns b_iP to $\mathscr{A}_{II}$ as the answer to the query submitting with ID_i.
- `Public Key Queries`: $\mathscr{F}_I$ submits queries with ID_i. The algorithm $\mathscr{B}$ calls private key oracle and searches the entry (ID_i, pk_i, x_i) in L_k, if found then $\mathscr{B}$ sends pk_i to $\mathscr{F}_I$. Else $\mathscr{B}$ chooses $x_i \in \mathbb{Z}_q^*$ at random and performs the following operations.
  1) If $ID_i = ID_l$, algorithm $\mathscr{B}$ answers $pk_l = x_iaP$ and if $ID_i = ID_\gamma$, $\mathscr{B}$ responses as $pk_\gamma = x_ibP$.
  2) If $ID_i \neq ID_l$ and If $ID_i \neq ID_\gamma$, algorithm constructs a new pairs (x_i, pk_i), where $pk_i = x_iP$ and adds the tuple (ID_i, pk_i, x_i) to L_k.
- H_2 Queries: $\mathscr{F}_I$ submits the queries with $(U, T, T_1, ID_B, pk_A, pk_B)$ to H_2 oracle. Algorithm $\mathscr{B}$ proceeds as follows:
  1) It searches the tuples $(U, T, *, ID_B, pk_A, pk_B, h)$ for distinct value of h, where $\hat{e}(x_iaP, x_ibP) = \hat{e}(x_i^2P, T_1)$. If there exists such tuples, it replace $*$ with T_1. Note that $ID_i = ID_l$ and $ID_i = ID_\gamma$.
  2) If $\mathscr{B}$ searches this point of execution, It generates a random h and updates the list L_2 with this tuples and the answer value h. Note that initially the list is empty.
- H_3 Queries: $\mathscr{F}_I$ submits queries with $(U, C_1, T_1, ID_A, pk_A)$ to H_3 oracle, algorithm $\mathscr{B}$ first searches the tuple in L_3, if founds return the answer, otherwise picks a random value t, returns tP and updates the list L_3.
- H_4 Queries: $\mathscr{F}_I$ submits queries with $(U, C_1, T_1, ID_A, pk_A)$ to H_4 oracle, algorithm $\mathscr{B}$ first searches the tuple in L_4, if founds return the answer, otherwise picks a random value w, returns wP and updates the list L_4.
- H_5 Queries: $\mathscr{F}_I$ submits queries with (U, T), algorithm $\mathscr{B}$ searches the list L_5 for the tuples (U, T, k), if it founds, then returns k to $\mathscr{F}_I$. Else $\mathscr{B}$ picks $k \in \{0, 1\}^n$ at random and replies this to $\mathscr{A}_{II}$ and updates L_5 with (U, T, k).
- `Private Key Queries`: $\mathscr{F}_I$ submits queries with (x_i, D_i), algorithm $\mathscr{B}$ calls H_1 with ID_i and proceeds as
  1) If $ID_i = ID_l$ or $ID_i = ID_\gamma$, $\mathscr{B}$ aborts the simulation and stop.
  2) If $ID_i \neq ID_l$, ID_γ , $\mathscr{B}$ searches the list L_k and returns $S_i = (x_i, D_i)$ to $\mathscr{F}_I$.
- `Proxy Key Construction Queries`: $\mathscr{F}_I$ submits queries with ID_i and ID_j. Algorithm $\mathscr{B}$ computes the delegater's private key S_i. It chooses randomly $\alpha, \beta \in \mathbb{Z}_q^*$ and the element W_2 from G_1 at random. Then sets $Z = \beta P_{pub} + x_iW_2, W_1 = \alpha P_{pub}$ and $U = (\beta P - Q_i)\alpha^{-1}$. Computes $R = r_1P, T^{'} = \hat{e}(D_j, R)$ and $T_2 = r_1pk_j$. $\mathscr{C}$ retrieves $\bar{W}$ from L_5 list and computes $X = \bar{W} \oplus W$. Finally this will return the proxy key $k_{ij} = (T_2, X)$ as answer. Consequently $\mathscr{C}$ fails to return the proxy key when $i = l$ or $i = \gamma$.
- `Signcryption Queries`: Let the sender and receiver's identities are denoted by ID_A and ID_B respectively. Let the m denotes the message on which the cipher is to be generated. $\mathscr{F}_I$ submits queries with the tuple (m, ID_A, ID_B). The algorithm $\mathscr{B}$ executes and answers in this ways. This has two cases.
  1) If $ID_A \neq ID_l$ and $ID_A \neq ID_\gamma$, algorithm $\mathscr{B}$ can obtain the private key S_A and computes σ_{AB}. Then $\mathscr{B}$ answers σ_{AB} to $\mathscr{A}_{II}$.
  2) If $ID_A = ID_l$ or $ID_A = ID_\gamma$, algorithm $\mathscr{B}$ chooses $r \in \mathbb{Z}_q^*$ at random and computes $U = rP, T = \hat{e}(D_B, U)$ and $T_1 = rpk_B$. Selects

the two elements W_1 and W_2 at random from G_1 and computes $W = W_1 \oplus W_2$. Then computes $Z = bP_{pub} + x_A W_2 + rW_1$. Sets $k = H_4(U, T)$ and defines $h = H_2(U, T, T_1, ID_B, pk_A, pk_B)$. Finally computes $C_1 = m \oplus h$ and $C_2 = W \oplus k$. The algorithm returns $\sigma_{AB} = (C_1, C_2, Z)$ to $\mathscr{F}_I$.

- `Re-encryption Queries`: $\mathscr{F}_I$ submits the queries with $(\sigma_{AB}, ID_A, ID_B, ID_C)$. Algorithm $\mathscr{B}$ proceeds on the following two cases:
  1) If $ID_B \neq ID_l, ID_\gamma$, It executes re-encryption and generates σ_{AC} through the queries submitted with (ID_B, ID_C).
  2) If $ID_B = ID_l$ or $ID_B = ID_\gamma$ and $ID_C \neq ID_l, ID_\gamma$, $\mathscr{B}$ chooses the element $T_2 \in G_1$, defines $W_1 = H_3(U, C_1, T_1, ID_B, pk_B)$ and $W_2 = H_4(U, C_1, T_1, ID_A, pk_A)$. Then computes $W = W_1 \oplus W_2$ and defines $\bar{W} = H_5(T', U, R, T_2, ID_B, pk_C)$. Finally computes $X = \bar{W} \oplus W$, compute $C_1' = C_1 + T_2$ and $C_2' = C_2 \oplus X$. So finally, the algorithm $\mathscr{B}$ returns $\sigma_{AC} = (C_1, C_1', C_2, Z)$ to $\mathscr{F}_I$.
- `Unsigncryption Queries`: $\mathscr{F}_I$ submits the queries on $\sigma_{AB} = (C_1, C_1', C_2, Z)$ with ID_A and ID_B. Algorithm $\mathscr{B}$ proceeds as:
  1) If $ID_B \neq ID_l$, then it obtains S_B by calling private key oracle with ID_B, runs Unsigncryption and returns m to $\mathscr{A}_{II}$.
  2) When $ID_B = ID_l$ or $ID_B = ID_\gamma$, then the pairing cannot be computed. In order to follow consistency in all answers, $\mathscr{B}$ searches the entry $(U, T, T_1, ID_B, pk_A, pk_B, h)$ for distinct value of T such that the decision bilinear Diffie-Hellman oracle returns 1 when the query submitted with (aP, bP, cP, U, T). If such an entry exists, then the corresponding value of U and T can be found. Algorithm $\mathscr{B}$ calls H_4 and obtains k. Then checks whether $W = C_2 \oplus k$ holds. If it fails, then $\mathscr{B}$ returns "0" and aborts the simulations. Otherwise, algorithm $\mathscr{B}$ calls H_2, obtains h, computes $C_1 \oplus h$ and recovers m. Finally to accept m, algorithm $\mathscr{B}$ checks if $\hat{e}(P, Z) = \hat{e}(P_{pub}, Q_A)\hat{e}(U, W_1)\hat{e}(pk_A, W_2)$ holds and returns m to $\mathscr{F}_I$, otherwise it returns "0" for invalid ciphertext.
  3) If $\mathscr{B}$ searches this point of execution, it chooses a random h on the list L_2 and places the entry $(U, *, T_1, ID_B, pk_B)$ and decrypts using this h as $C_1 \oplus h$. Note that the symbol $*$ denotes the unknown value of pairing. Finally verifies if $\hat{e}(P, Z) = \hat{e}(P_{pub}, Q_A)\hat{e}(U, W_1)\hat{e}(pk_A, W_2)$ holds and returns m to $\mathscr{F}_I$. Otherwise it returns "0" for invalid ciphertext.
- `Decryption Queries`: $\mathscr{A}_{II}$ submits the queries on $\sigma_{AC} = (C_1, C_1', C_2, Z)$ with ID_A, ID_B and ID_C. Algorithm $\mathscr{B}$ proceeds as:
  1) If $ID_C \neq ID_l$, it requests private key oracles and obtains S_C. Then computes m and returns to $\mathscr{F}_I$.
  2) If $ID_C = ID_l$, then $\mathscr{B}$ goes to L_5 and looks for the tuple $(T_\xi', U, R_\xi, T_{2\xi}, ID_B, pk_C, \bar{W}_\xi)$ indexed by $\xi \in \{1, 2 \ldots q_{H_5}\}$. The algorithm $\mathscr{B}$ takes T_2 and computes $C_1 = C_1' - T_2$ for $T_2 = T_{2\xi}$. If $T_2 \neq T_{2\xi}$ then the algorithm $\mathscr{B}$ looks for next element of L_5, else computes $m = C_1 \oplus h$ and $k = C_2' \oplus \bar{W}_\xi$). The algorithm $\mathscr{B}$ defines $h = H_2(U, T, T_1, ID_B, pk_A, pk_B)$ and checks if $\hat{e}(P, Z) = \hat{e}(P_{pub}, Q_A)\hat{e}(U, W_1)\hat{e}(pk_A, W_2)$ holds. Then returns m to $\mathscr{F}_I$. Else $\mathscr{B}$ looks next element of L_5, if it does not found such message m in the complete searching, then it returns "0" and terminates the simulation.
- `Challenge`: Eventually, $\mathscr{F}_I$ returns two messages (m_0, m_1) of equal length. and two identities ID_A^* and ID_B^*. Algorithm $\mathscr{B}$ looks the list L_1 with input ID_B^*. If $ID_B^* \neq ID_l$, then $\mathscr{B}$ fails and aborts the simulation. Otherwise it proceeds to generate a challenge as follows:
  1) $\mathscr{B}$ looks the public key pk_A^* corresponding to ID_A^* in L_k. Then it sets $U^* = cP$, chooses the elements $T_1^*, W_1^*, W_2^* \in G_1$ at random and $k^* \in \{0,1\}^n$. Then a hash value h^* is chosen and $W^* = W_1^* \oplus W_2^*$is computed.
  2) $\mathscr{B}$ picks a random bit $\varphi \in \{0, 1\}$ and sets $C_1^* = m_\varphi \oplus h^*$, $C_2^* = W^* \oplus k^*$.
  3) The component Z^* is set to $D_A + rW_1 + x_A W_2 = D_A + \omega cP + \tau pk_A^*$. Where $\mathscr{B}$ takes ω and τ from L_3 and L_4 respectively. D_A is computed by calling partial private key extraction oracle on ID_A^*. Note that, since $ID_A^* \neq ID_B^*$, the partial private key extraction oracles returns correct value of D_A. Finally $\mathscr{B}$ returns $\sigma_{AB}^* = (C_1^*, C_2^*, Z^*)$ to $\mathscr{F}_I$.
- `Phase-II`: As in Phase-I, $\mathscr{F}_I$ submits polynomial number of queries in an adaptive manner and the algorithm $\mathscr{B}$ answers as a response to all queries to $\mathscr{F}_I$.
- `Guess`: Eventually, $\mathscr{F}_I$ returns its guess as to which message is signcrypted during the challenge. The adversary returns an identity ID_B^* with index l, since l is independent of adversary's view with probability $\frac{1}{q_{H_1}}$. If this event occurs, the simulation would be success unless it submits the queries with challenged-related tuples $(U^*, T^*, T_1^*, ID_B^*, pk_A^*, pk_B^*)$ to H_2 oracle. Where $T_1 = x_i^2 abP, pk_A^* = x_i aP$ and $pk_B^* = x_i bP$. Note that H_2 is modelled as random oracle, if the tuples does not belong to L_2 list, the adversary does not have any advantage. The algorithm $\mathscr{B}$ checks the following equation

$$\hat{e}(x_i^2 P, T_1) = \hat{e}(x_i aP, x_i bP)$$

If it holds then the algorithm aborts and stops the simulation. This returns the output $T_1^* = abP$. Thus one of the it would be answer of CDH problem. Thus $\mathscr{B}$ returns wins the IND-iCCA-II game solving **CDH** problem in the simulation of H_2 .

`Attack`: The algorithm sets challenged identity $ID_A = ID_l$ in H_1 oracle.

`Forgery`: $\mathscr{F}_I$ outputs a signcryption σ_{AB}^*, where $\sigma_{AB}^* = (C_1^*, C_2^*, Z^*)$. Here the Alice's and Bob's identity are ID_A^*

and ID_B^*. Algorithm $\mathscr{B}$ calls H_1 oracle on ID_A^* and checks if $ID_A^* \neq ID_l$, then aborts the simulation and stops. Else, $\mathscr{B}$ calls public key oracles on ID_A and obtains pk_A^*. Takes t^* and w^* from L_3 and L_4 respectively by submitting queries on $(U^*, C_1^*, T_1^*, ID_l, pk_A^*)$. Note that $\mathscr{F}_I$ wins if the following verification equation holds.

$\hat{e}(P, Z^*) = \hat{e}(P_{pub}, Q_l)\hat{e}(U^*, W_1^*)\hat{e}(pk_l^*, W_2^*)$
$\hat{e}(P, Z^*) = \hat{e}(aP, bP)\hat{e}(U^*, t^*P)\hat{e}(pk_l^*, w^*P).$
$\hat{e}(P, Z^*) = \hat{e}(P, abP)\hat{e}(P, t^*U^*)\hat{e}(P, w^*pk_l^*).$
This implies $Z^* = abP + t^*U^* + w^*pk_l^*.$
$\implies abP = Z^* - t^*U^* - w^*pk_l^*.$

By applying forking lemma [28] $\mathscr{F}_I$ successes in time t with probability $\epsilon_{cdh} \geq \frac{10(q_s+1)(q_s+q_{H_3}+q_{H_4})q_{H_1}}{2^{\psi}-1}$, then $\mathscr{C}$ can solve the mathematical hard CDH problem in a time $t^{'} \leq \Psi q_{H_1} q_{H_3} q_{H_4}(t + \mathcal{O}(q_{pk} + q_s + q_e + q_u + q_d)t_p/\epsilon(1 - \frac{1}{2^{\psi}})$, where $\Psi = 120686$ and t_p denotes the pairing cost. □

Proof of Lemma: 4

Proof. The simulation shows, how the CDH problem is solved on the forgery. The algorithm $\mathscr{B}$ uses $\mathscr{F}_{II}$ and proceeds as:

- `Setup`: Algorithm $\mathscr{B}$ takes the CDH instances (P, aP, bP). This sets $P_{pub} = sP$ and provides P_{pub} with system parameters $param$. Where s is considered as master secret key.
- H_1 Queries: Algorithm $\mathscr{B}$ selects an index l uniformly at random in $\{1, 2 \dots q_{H_1}\}$. Assume that $\mathscr{F}_{II}$ submits queries to H_1-oracle on distinct identities ID_i. Where i sets to counter 1. $\mathscr{F}_{II}$ does not submit the query on an identity , if it has been queried and answers the stored value in L_1 for that particular identity as described in lemma 1. $\mathscr{F}_{II}$ chooses (ID_l, ID_γ) as challenge identities from the set $\{1, 2 \dots q_{H_1}\}$. The algorithm $\mathscr{B}$ chooses $b_i \in \mathbb{Z}_q^*$ at random, and sets $H_1(ID_i) = b_iP$, then adds (ID_i, b_iP, b_i) to list L_1 which is initially empty and returns b_iP to $\mathscr{F}_{II}$ as answer to the query submitting with ID_i.
- `Public Key Queries`: $\mathscr{F}_{II}$ submits queries with ID_i. The algorithm $\mathscr{B}$ calls private key oracle and searches the entry (ID_i, pk_i, x_i) in L_k, if found then $\mathscr{B}$ sends pk_i to $\mathscr{F}_{II}$. Else $\mathscr{B}$ chooses $x_i \in \mathbb{Z}_q^*$ at random and performs the following
  1) If $ID_i = ID_l$, algorithm $\mathscr{B}$ answers $pk_l = x_iaP$ and if $ID_i = ID_\gamma$, $\mathscr{B}$ responses as $pk_\gamma = x_ibP$.
  2) If $ID_i \neq ID_l$ and If $ID_i \neq ID_\gamma$, algorithm constructs a new pairs (x_i, pk_i), where $pk_i = x_iP$ and adds the tuple (ID_i, pk_i, x_i) to L_k.
- H_2 Queries: $\mathscr{A}_{II}$ submits the queries with $(U, T, T_1, ID_B, pk_A, pk_B)$ to H_2 oracle. Algorithm $\mathscr{B}$ proceeds as follows:
  1) It searches the tuples $(U, T, *, ID_B, pk_B, h)$ for distinct value of h, where $\hat{e}(x_iaP, x_ibP) = \hat{e}(x_i^2P, T_1)$. If there there exists such tuple, it replaces $*$ with T_1. Note that $ID_i = ID_l$ and $ID_i = ID_\gamma$.
  2) If $\mathscr{B}$ searches this point of execution, It generates a random h and updates the list L_2 with this tuples and the answer value h. Note that initially the list is empty.
- H_3 Queries: $\mathscr{A}_{II}$ submits queries with $(U, C_1, T_1, ID_A, pk_A)$ to H_3 oracle, algorithm $\mathscr{B}$ first searches the tuple in L_3, if founds return the answer, otherwise picks a random value t, returns tP and updates the list L_3.
- H_4 Queries: $\mathscr{A}_{II}$ submits queries with $(U, C_1, T_1, ID_A, pk_A)$ to H_4 oracle, algorithm $\mathscr{B}$ first searches the tuple in L_4, if founds return the answer, otherwise picks a random value w, returns wP and updates the list L_4.
- H_5 Queries: $\mathscr{A}_{II}$ submits queries with (U, T), algorithm $\mathscr{B}$ searches the list L_5 for the tuples (U, T, k), if it founds, then return k to $\mathscr{A}_{II}$. Else $\mathscr{B}$ picks $k \in \{0, 1\}^n$ at random and replies this to $\mathscr{A}_{II}$ and updates L_5 with (U, T, k).
- `Private Key Queries`: $\mathscr{A}_{II}$ submits queries with (x_i, D_i), algorithm $\mathscr{B}$ calls H_1 with ID_i and proceeds as
  1) If $ID_i = ID_l$ or $ID_i = ID_\gamma$, $\mathscr{B}$ aborts the simulation and stop.
  2) If $ID_i \neq ID_l$, ID_γ , $\mathscr{B}$ searches the list L_k and returns $S_i = (x_i, D_i)$ to $\mathscr{A}_{II}$.
- `Proxy Key Construction Queries`: $\mathscr{A}_{II}$ submits queries with ID_i and ID_j. Algorithm $\mathscr{B}$ computes the delegater's private key S_i. It chooses randomly $\alpha, \beta \in \mathbb{Z}_q^*$ and the element W_2 from G_1 at random. Then sets $Z = \beta P_{pub} + x_iW_2, W_1 = \alpha P_{pub}$ and $U = (\beta P - Q_i)\alpha^{-1}$. Computes $R = r_1P$, $T^{'} = \hat{e}(D_j, R)$ and $T_2 = r_1pk_j$. $\mathscr{C}$ retrieves $\bar{W}$ from L_5 list and computes $X = \bar{W} \oplus W$. Finally this returns the proxy key $k_{ij} = (T_2, X)$ as answer. Consequently $\mathscr{C}$ fails to return the proxy key when $i = l$ or $i = \gamma$.
- `Signcryption Queries`: Let the sender and receiver's identities are denoted by ID_A and ID_B respectively. Let the m denotes the message on which the cipher is to be generate. $\mathscr{A}_{II}$ submits queries with the tuple (m, ID_A, ID_B). The algorithm $\mathscr{B}$ executes and answers in this ways. This is having two cases.
  1) If $ID_A \neq ID_l$ and $ID_A \neq ID_\gamma$, algorithm $\mathscr{B}$ can obtain the private key S_A and computes σ_{AB}. Then $\mathscr{B}$ answers σ_{AB} to $\mathscr{A}_{II}$.
  2) If $ID_A = ID_l$ or $ID_A = ID_\gamma$, algorithm $\mathscr{B}$ chooses $r \in \mathbb{Z}_q^*$ at random and computes $U = rP, T = \hat{e}(D_B, U)$ and $T_1 = rpk_B$. Selects the two elements W_1 and W_2 at random from G_1 and computes $W = W_1 \oplus W_2$. Then computes $Z = bP_{pub} + x_AW_2 + rW_1$. Sets $k = H_4(U, T)$ and defines $h = H_2(U, T, T_1, ID_B, pk_A, pk_B)$. Finally computes $C_1 = m \oplus h$ and $C_2 = W \oplus k$. The algorithm returns $\sigma_{AB} = (C_1, C_2, Z)$ to $\mathscr{A}_{II}$.
- `Re-encryption Queries`: $\mathscr{A}_{II}$ submits the queries with $(\sigma_{AB}, ID_A, ID_B, ID_C)$. Algorithm $\mathscr{B}$ proceeds on the following two cases:
  1) If $ID_B \neq ID_l, ID_\gamma$, It executes re-encryption and generates σ_{AC} through the queries submitted with (ID_B, ID_C).

2) If $ID_B = ID_l$ or $ID_B = ID_\gamma$ and $ID_C \neq ID_l, ID_\gamma$, $\mathscr{B}$ chooses the element $T_2 \in G_1$, defines $W_1 = H_3(U, C_1, T_1, ID_B, pk_B)$ and $W_2 = H_4(U, C_1, T_1, ID_A, pk_A)$. Then computes $W = W_1 \oplus W_2$ and defines $\bar{W} = H_5(T^{'}, U, R, T_2, ID_B, pk_C)$. Finally computes $X = \bar{W} \oplus W$, $C_1^{'} = C_1 + T_2$ and $C_2^{'} = C_2 \oplus X$. So finally, the algorithm $\mathscr{B}$ returns $\sigma_{AC} = (C_1, C_1^{'}, C_2, Z)$ to $\mathscr{A}_{II}$.

- `Unsigncryption Queries`: $\mathscr{A}_{II}$ submits the queries on $\sigma_{AB} = (C_1, C_1^{'}, C_2, Z)$ with ID_A and ID_B. Algorithm $\mathscr{B}$ proceeds as:
  1) If $ID_B \neq ID_l$, then it obtains S_B by calling private key oracle with ID_B, runs Unsigncryption and returns m to $\mathscr{A}_{II}$.
  2) When $ID_B = ID_l$ or $ID_B = ID_\gamma$, then the pairing cannot be computed. In order to follow consistency in all answers, $\mathscr{B}$ searches the entry $(U, T, T_1, ID_B, pk_A, pk_B, h)$ for distinct value of T such that the decision bilinear Diffie-Hellman oracle returns 1 when the query submitted with (aP, bP, cP, U, T). If such an entry exists, then the corresponding value of U and T can be found. Algorithm $\mathscr{B}$ calls H_4 and obtains k. Then checks whether $W = C_2 \oplus k$ holds. If it false then $\mathscr{B}$ returns "0" and aborts the simulations. Otherwise, algorithm $\mathscr{B}$ calls H_2, obtains h, computes $C_1 \oplus h$ and recovers m. Finally to accept m, algorithm $\mathscr{B}$ checks if $\hat{e}(P, Z) = \hat{e}(P_{pub}, Q_A)\hat{e}(U, W_1)\hat{e}(pk_A, W_2)$ holds and returns m to $\mathscr{A}_I$, otherwise it returns "0" for invalid ciphertext.
  3) If $\mathscr{B}$ searches this point of execution, it chooses a random h on the list L_2 and places the entry $(U, *, T_1, ID_B, pk_B)$ and decrypts using this h as $C_1 \oplus h$. Note that the symbol $*$ denotes the unknown value of pairing. Finally verifies if $\hat{e}(P, Z) = \hat{e}(P_{pub}, Q_A)\hat{e}(U, W_1)\hat{e}(pk_A, W_2)$ holds and returns m to $\mathscr{A}_I$. Otherwise it returns "0" for invalid ciphertext.
- `Decryption Queries`: $\mathscr{A}_{II}$ submits the queries on $\sigma_{AC} = (C_1, C_1^{'}, C_2, Z)$ with ID_A, ID_B and ID_C. Algorithm $\mathscr{B}$ proceeds as:
  1) If $ID_C \neq ID_l$, it requests private key oracles and obtains S_C. Then computes m and returns to $\mathscr{A}_I$.
  2) If $ID_C = ID_l$, then $\mathscr{B}$ goes to L_5 and looks for the tuple $(T_\xi^{'}, U, R_\xi, T_{2\xi}, ID_B, pk_C, \bar{W}_\xi)$ indexed by $\xi \in \{1, 2 \ldots q_{H_5}\}$. The algorithm $\mathscr{B}$ takes T_2 and computes $C_1 = C_1^{'} - T_2$ for $T_2 = T_{2\xi}$. If $T_2 \neq T_{2\xi}$ then the algorithm $\mathscr{B}$ looks for next element of L_5, else computes $m = C_1 \oplus h$ and $k = C_2^{'} \oplus \bar{W}_\xi$). The algorithm $\mathscr{B}$ defines $h = H_2(U, T, T_1, ID_B, pk_A, pk_B)$ and checks if $\hat{e}(P, Z) = \hat{e}(P_{pub}, Q_A)\hat{e}(U, W_1)\hat{e}(pk_A, W_2)$ holds. Then returns m to $\mathscr{A}_I$. Else $\mathscr{B}$ looks next element of L_5, if it does not found such message m in the complete searching, then it returns "0" and terminates the simulation.
- `Attack`: $\mathscr{F}_{II}$ submits polynomially bounded number of queries as in `Phase-I` of `Game-II`.
- `Forgery`: Eventually, $\mathscr{F}_{II}$ returns the output a signcryption σ_{AB}^*, where $\sigma_{AB}^* = (C_1^*, C_2^*, Z^*)$. Here the Alice's and Bob's identity are ID_A^* and ID_B^*. Algorithm $\mathscr{B}$ calls public key oracle on ID_A^* and checks if $ID_A^* \neq ID_l$, then aborts the simulation and stop. Else, $\mathscr{B}$ calls public key oracles on ID_A and obtains pk_A^*. Takes t^* and w^* from L_3 and L_4 respectively by submitting queries on $(U^*, C_1^*, T_1^*, ID_l, pk_A^*)$. Note that $\mathscr{F}_{II}$ wins if the following verification equation holds.
  $\hat{e}(P, Z^*) = \hat{e}(P_{pub}, Q_l)\hat{e}(U^*, W_1^*)\hat{e}(pk_l^*, W_2^*)$
  $\hat{e}(P, Z^*) = \hat{e}(sP, Q_l)\hat{e}(U^*, t^*P)\hat{e}(aP, w^*bP)$.
  $\hat{e}(P, Z^*) = \hat{e}(P, sQ_l)\hat{e}(P, t^*U^*)\hat{e}(aP, w^*bP)$.
  This implies $Z^* = D_l + t^*U^* + w^*abP$.
  $\implies w^*abP = Z^* - D_l - t^*U^*$
  $\implies abP = (Z^* - D_l - t^*U^*){w^*}^{-1}$

□