




Please cite the Published Version

Unal, Devrim , Al-Ali, Abdullah, Catak, Ferhat Ozgur  and Hammoudeh, Mohammad  (2021)
A secure and efficient Internet of Things cloud encryption scheme with forensics investigation compatibility based on identity-based encryption. Future Generation Computer Systems, 125. pp. 433-445. ISSN 0167-739X

DOI: <https://doi.org/10.1016/j.future.2021.06.050>

Publisher: Elsevier

Version: Accepted Version

Downloaded from: <https://e-space.mmu.ac.uk/634406/>

Usage rights:  [Creative Commons: Attribution-Noncommercial-No Derivative Works 4.0](https://creativecommons.org/licenses/by-nc-nd/4.0/)

Additional Information: © 2021. This manuscript version is made available under the CC-BY-NC-ND 4.0 license <https://creativecommons.org/licenses/by-nc-nd/4.0/>

Enquiries:

If you have questions about this document, contact openresearch@mmu.ac.uk. Please include the URL of the record in e-space. If you believe that your, or a third party's rights have been compromised through this document please see our Take Down policy (available from <https://www.mmu.ac.uk/library/using-the-library/policies-and-guidelines>)

A secure and efficient Internet of Things cloud encryption scheme with forensics investigation compatibility based on identity-based encryption

Devrim Unal*

KINDI Center for Computing Research, College of Engineering, Qatar University, Doha, Qatar

Abdulla Al-Ali

Department of Computer Science and Engineering, College of Engineering, Qatar University, Doha, Qatar

Ferhat Ozgur Catak

SIMULA Research Laboratory, Fornebu, Norway

Mohammad Hammoudeh

Department of Computing and Mathematics, Manchester Metropolitan University, Manchester, M15 6BH, United Kingdom

Abstract

Data security is a challenge for end-users of cloud services as the users have no control over their data once it is transmitted to the cloud. A potentially corrupt cloud service provider can obtain the end-users' data. Conventional PKI-based solutions are insufficient for large-scale cloud systems, considering efficiency, scalability, and security. In large-scale cloud systems, the key management requirements include scalable encryption, authentication, and non-repudiation services, as well as the ability to share files with different users and data recovery when the user keys of encrypted data are not accessible. Further requirements in cloud systems include the ability to provide the means for digital forensic investigations on encrypted data. Once data on the cloud is encrypted with a user's key it becomes impossible to access by forensic investigation teams. In this regard, distributing the trust of key management into multiple authorities is desirable. In the literature, there is no available secure cloud storage system with secure and efficient Type-3 pairings, supporting Encryption-as-a-Service (EaaS) and multiple Public Key Generators (PKGs). This paper proposes an efficient Identity-based cryptography (IBC) architecture for secure cloud storage, named Secure Cloud Storage System (SCSS), which supports distributed key management and encryption mechanisms and support for multiple PKGs. During forensic investigations, the legal authorities will be able to use the multiple PKG mechanism for data access, while an account locking mechanism prevents a single authority to access user data due to trust distribution. We also demonstrate that, the IBC scheme used in SCSS has better performance compared to similar schemes in the literature. For the security levels of 128-bits and above, SCSS has better scalability compared to existing schemes, with respect to encryption and decryption operations. Since the decryption operation is frequently needed for forensic analysis, the improved scalability results in a streamlined forensic investigation process on the encrypted data in the cloud.

Keywords: Type-3 Pairings, Cloud Security, Encryption-as-a-Service, Identity-based Cryptography

2010 MSC: 00-01, 99-00

1. Introduction

Cloud storage services not only attract tremendous attention as means to store, access and manage data on a pay per use basis, but they also bring cost reduction and ease of management benefits to enterprises. According to a recent report by Mordor Intelligence [1], The global cloud storage market is

expected to reach USD 170.02 billion by 2025. In the Internet of Things(IoT) systems, the storage of data on the cloud is widely applied due to the large volume of information generated by IoT devices and the need to share information between multiple parties. Figure 1 depicts an example context of cloud storage services for IoT based systems. In this example, sensors and smart devices generate data that are transmitted to the cloud.

Typically, all cloud participating entities have access to an Encryption-as-a-Service (EaaS) for individual, organization and aggregated data based on their access rights. EaaS provides access control as well as encryption on the cloud. In this scenario,

*This study has been funded by the Qatar National Research Fund (QNRF), with grant number NPRP10-0125-170250

*Corresponding author

Email addresses: dunal@qu.edu.qa (Devrim Unal), abdulla.alali@qu.edu.qa (Abdulla Al-Ali), ozgur@simula.no (Ferhat Ozgur Catak), m.hammoudeh@mmu.ac.uk (Mohammad Hammoudeh)

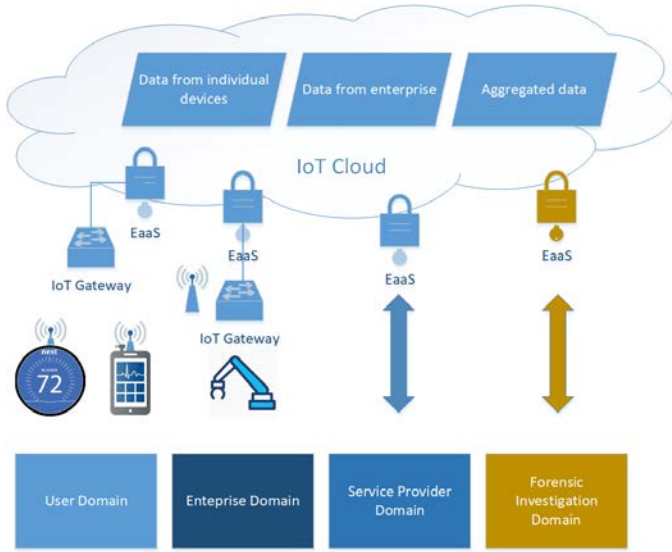


Figure 1: IoT secure cloud encryption with Encryption-as-a-Service

the generation and management of the keys become a critical task. Since the data is in a central location, symmetric encryption is not feasible. If one of the parties loses the symmetric key, unauthorized parties could access the whole data on the cloud. Therefore, in an IoT-based cloud storage setting, a scalable asymmetric key management solution where the keys are bound to the identity of the entities is required.

Despite of the cloud's attractive benefits, several security issues arise due to the outsourcing of storage services, virtualization, and multi-tenancy. Users are mainly concerned about the loss of control over their data and Cloud Service Provider's (CSP) viability and operational processes. Users of these services face the risks of stolen, erased, or lost data. Therefore, the security of data at rest and security of data in transfer must be considered when designing and deploying cloud-based storage services [2]. CSPs have to provide trustworthy and secure cloud storage to ensure the confidentiality and integrity of data [3]. Moreover, due to multi-tenancy, multiple users' data may be located in the same physical machine, and malicious adversaries may take advantage of data co-location to break the confidentiality of the overall system. Information leakage may also occur due to system errors or vulnerabilities in the infrastructure software.

Over recent years, there is a growing requirement for outsourced cloud systems that allow tenants to rapidly manage sensitive data collected from systems, including health care IoT or financial systems. Additionally, public big data systems are required to investigate data without violating data privacy. Deep learning is a new age that can infer invaluable information to reach decisions efficiently using different public cloud systems. Sometimes, cloud platforms include data coupled with other privacy policies; therefore, they should be subject to security analysis. Suppose the case of different medical institutes that want to jointly build a virus diagnosis system using a deep learning algorithm. In this case, privacy policies and General Data Protection Regulation (GDPR) blocks these institutes from sharing the patients' medical records. Conventional deep learn-

ing methods can not be applied and sensitive medical data records can not be shared publicly due to the different privacy policies of other parties. Hence, an identity-based cryptography (IBC) that are compliant with privacy policies of various organizations is needed.

Many data security schemes suffer from the fact that the keys for encrypting the data resides with either the user or the CSP. In either case, when there is a legal court order that requires access to the encrypted data, or an enterprise-wide forensic investigation, the investigation authorities do not have the means to decrypt the data without the user consent. While privacy is desired for regular operations, there are certain cases which involve cyber-attacks or insider breaches, in which the digital forensic investigation team would need access to encrypted data. Traditional digital forensics is an applied science for identifying, collecting, organizing, and presenting digital evidence. Cloud forensics includes these four methods in the cloud, where evidence covers stored files, process and network logs, and registry records of Virtual Machines (VMs) working in the cloud. Conventional forensics assumes physical access to evidence, which is not viable in the cloud. In cloud forensics, identifying, collecting, organizing and presenting digital evidence mainly depends on cloud providers' infrastructure. The could systems are designed as a black-box environment. Forensic investigators can not prove the legality of the collected digital evidence or guarantee its integrity [4].

In this work, a novel secure cloud storage solution is proposed, named Secure Cloud Storage System (SCSS), which is based on IBC and provides encryption-as-a-service (EaaS). The proposed scheme is compatible with digital forensics investigations in that a multiple Public Key Generator (PKG) based secret sharing scheme is utilized to generate the keys. In this way, legal authorities may act as a party in the key generation in conjunction with another trusted key generation authority which acts as the other PKG. When a decryption of an encrypted file is required, the legal authority collaborates with the trusted key generation authority to re-generate the private key and decrypt the file contents and provide the decrypted files for digital forensics investigations. Neither the legal authority nor the trusted key generation authority are able to generate the private key by themselves, thus preventing the misuse of this mechanism.

The SCSS aims at providing efficient, secure, and scalable data storage in the cloud. Our solution is easier to apply and more efficient than existing secure cloud storage solutions, especially considering key management and data sharing, which eliminates the need for resource-consuming certificate management and client-side key management. Furthermore, our proposal is the first solution providing both Identity-based cryptography (IBC) and service-oriented features [5]. Identity-Based Encryption (IBE) is the most important application of IBC and is the encryption scheme utilized in the proposed SCSS system. IBE is a type of public-key encryption in which the public key of a user is unique information about his/her identity.

The main contributions of the proposed work can be summarized as follows: SCSS is based on Type-3 pairings and is more secure and efficient than the existing Type-1 pairing-based

IBE schemes. SCSS includes a multiple-PKG key management scheme adopted for Type-3 pairing based SAKKE-IBE scheme, which is the first in literature. In this regard, SCSS is the first IBE based secure cloud storage system which supports multiple PKGs. The use of multiple PKGs can also be utilized in digital forensic use cases on the cloud where investigation authorities need to access encrypted data for forensic investigations. The distributed PKG works in a network of n nodes with a t -limited Byzantine adversary, i.e., in order to compromise of a user in the normal operation, at least t PKG nodes have to collaborate out of a total of n nodes.

We take into consideration, the Generic Computer Forensic Investigation Model (GCFIM), proposed by Yusoff et al. [6]. This model consists of 5 phases (Pre-Process, Acquisition & Preservation, Analysis, Presentation and Post-Process). SCSS can be used in the first two phases. In the pre-process phase, tasks prior to the actual investigation and official collection of data are conducted. In the Acquisition & Preservation phase all relevant data for the forensic investigation are captured. The secure cloud storage system proposed will be used in the secure data search and retrieval in the mentioned first two phases of the generic model.

Additionally, the proposed solution outperforms existing IBE based cloud security solutions based on analytical and experimental results presented in this paper. The approach introduced in this paper is more efficient than the existing IBE schemes since only the symmetric encryption key is encrypted by IBE, and the actual file is encrypted using symmetric encryption. An experimental study is introduced comparing Type-1 IBE [7], ABE [9, 10, 11, 12], and HIBE [13] schemes to the proposed SCSS scheme and show that our proposal provides better performance than existing schemes. The private key generation, encryption, and decryption times of SCSS are much lower than those compared schemes, and this is the basis of the scalability of SCSS. The number of private key generations increases with the number of users, whereas, the number of IBE based encryption/decryption operations increases with the number of file encryption/decryptions. By providing a faster cryptographic scheme which increases the performance for all of these operations, the basis for a scalable IBE-based cloud secure storage architecture is established. SCSS introduces IBE to the area of service-oriented cloud encryption solutions without compromising from security and performance.

2. Materials and Methods

2.1. Background and Related Work

Traditionally, cryptographic solutions for the cloud have been based on Public-Key Infrastructure (PKI). The main difference between PKI and IBC is the method for generation, binding, and verification of public-private key pairs, all without including a digital signature, as discussed in [14] and [15]. IBE is based on bilinear pairings on elliptic curves, it provides better security at smaller key sizes [16]. IBC utilizes Elliptical Curve Cryptography (ECC) in contrast to traditional PKI based systems that use the RSA algorithm. ECC provides stronger security and better server utilization than RSA with a shorter key

length. A benchmarking report published by Symantec [17] demonstrated the advantage of using ECC based cryptographic protocols to provide scalable cloud-based security solutions. Modified ECC algorithm has been utilized in [18] to provide secure de-duplication for file storage in cloud systems.

IBC avoids trust problems encountered in certificate-based PKIs. In PKI, the key pair is generated from random information, which results in a requirement for a certificate to bind the public key to its particular user. In contrast, the key pair in an IBC environment is generated explicitly from a user's publicly available information. This information could be the identity of the user in the system. Additionally, PKI-based solutions require Certificate Revocation Lists (CRL), which become complex and costly to manage for many enterprises. IBC does not require CRLs, and using IBC in a cloud environment reduces key management complexity and bandwidth utilization when compared to a certificate-based approach [9]. According to [19], there are various cloud encryption solutions; however, there is yet no cloud encryption solution that supports IBC. There is a secure e-mail solution from Micro Focus [20] that employs IBC which is a showcase of an IBC-based security product.

Figure [?] shows the various types of IBE as per the classification of Boyen [21] and various extensions. The idea behind IBE is to use identity attributes, such as ID numbers, e-mail addresses, or phone numbers, as the users' unique public keys. This property of a cryptographic system reduces the complexity of generating and deploying of users' certificates and does not need a conventional certification authority. Boneh and Franklin [22], and Cocks [23] proposed IBE schemes, independently, which both are based on bilinear maps of certain elliptic curves. Sakai-Kasahara proposed the SAKKE-IBE scheme [24], which has a different key extraction mechanism to reduce complexity. SAKKE-IBE has the advantage that the costly pairing operations are only needed in the decryption process. This new scheme has the advantage that it does not need a secure hashing map to elliptic curve elements but rather to an element in Z_q^* which is not only much more efficient but also realizable in an easy manner. There are a small number of cryptographic schemes that are based on IBE targeting the cloud environment for secure cloud storage. The existing works have some efficiency and security shortcomings which are summarized in Table 1.

Our extended proposal, named SCSS-SAKKE-IBE, presents a more secure adaptation of the IBC concept for the cloud compared to existing work. More specifically, Type-3 pairings were used; instead of Type-1 pairings as used in [9, 11, 7, 12, 8], which are considered to have security deficiencies [25]. The SCSS-SAKKE-IBE is more efficient than the other IBE schemes [9, 7, 12, 8] according to the simulation and experimental studies given in this paper. In comparison to [10] and [11], the keys of SCSS users can not be revoked without their consent and a user's identity is proved in a distributed manner, where the user is also part of the key establishment protocol, which provides more user control over existing works for data access. The user keys are re-generated after a forensic investigation so that the SCSS user can have continued and secure access to the cloud.

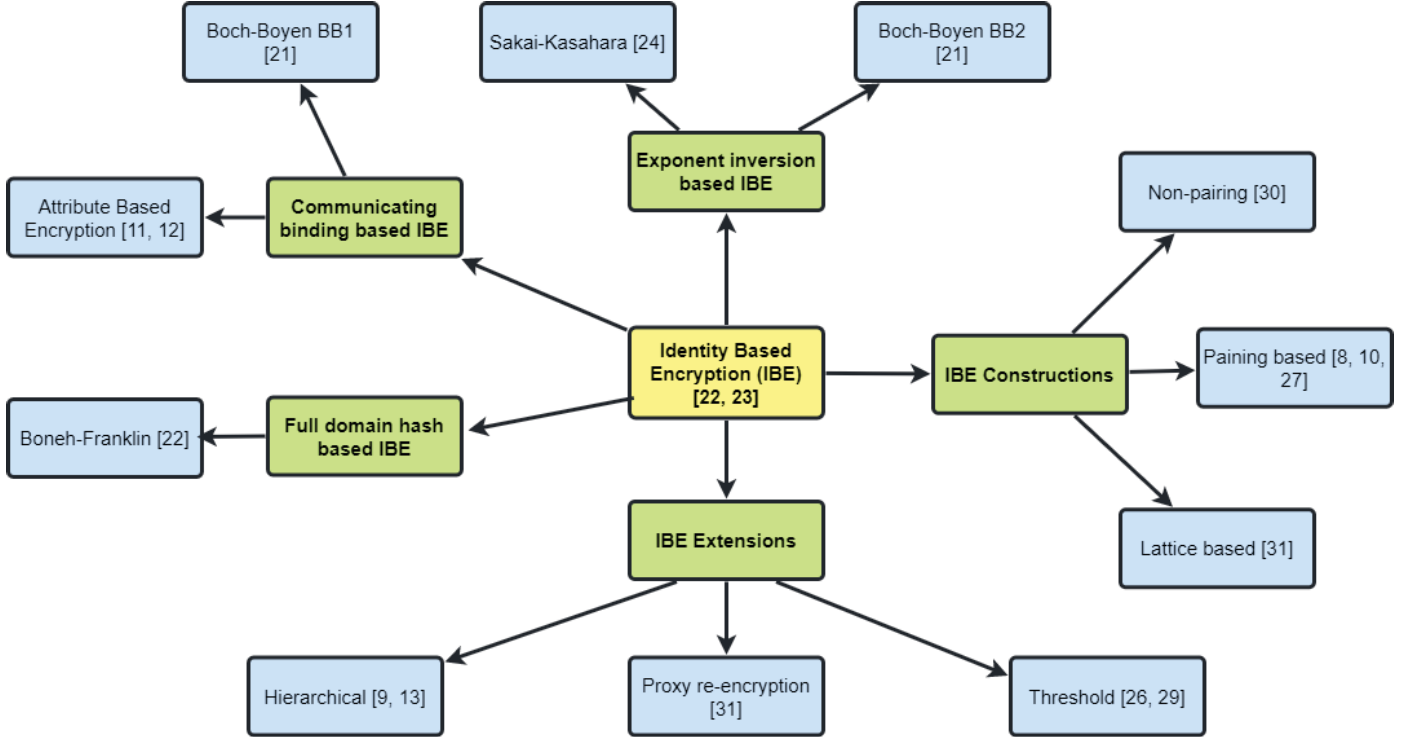


Figure 2: Identity Based Encryption schemes

Table 1: Identity-based cryptographic schemes for secure cloud storage

Scheme	Cryptographic Model	Key Management	Security	Multiple PKGs	EaaS
[9]	IBE & PKI	Delegated/ Certificate based	RSA encryption & Type-1 pairings, up to 128-bit	No	No
[10]	ABE	Distributed	Up to $N-2$ TA collusions, 80 bits	No	No
[11]	ABE	Hierarchical-Centralized	Type-1 pairings, up to 128-bit	No	No
[7]	IBE (BF)	User-centric	Type-1 pairings, up to 128-bit	No	No
[12]	HIBE	Centralized	Type-1 pairings, up to 128-bit	No	Yes
[8]	IBE (BF)	Centralized	Type-1 pairings, up to 128-bit	No	No
SCSS- SAKKE- IBE	IBE (SAKKE)	Distributed	Type-3 pairings, 192 bit and above	Yes	Yes

A multiple-PKG key management scheme is also included to the SCSS-SAKKE-IBE scheme, which is the first in the literature for Type-3 IBE schemes.

In single PKG scenarios, as the PKG computes a private key for a client, it can decrypt all messages passively. In our scheme, the extraction of the private Key takes places according to the protocol outlined in Section 3.2. In this protocol, the user key can not be extracted without the involvement of the minimum number of PKGs (t out of n). In an (n, t) -distributed PKG, the master key is distributed among n PKG nodes, if t or less number of nodes collaborate, they can not compute the master key. A user extracts the private key by obtaining private key shares from a minimum of $t + 1$ nodes. A user can verify the correctness of the extracted key with the system's public key. A multiple-PKG solution for IBE is presented in [26]; however, this is limited to Type-1 and Type-2 pairings. There is a recent extension of SAKKE-IBE to Type-3 pairings [27]; however, this work does not include multiple PKGs. Moreover, this proposal provides support for multiple PKGs. Recently the multiple PKG approach has been applied to Single

sign-on (SSO) [28] and key issuing [29]; however, there is no secure cloud storage solution supporting multiple PKGs in the literature.

In [9], the authors claim an IBC-based cryptographic model; however, the secure channel between the service and the private key generator is achieved through RSA encryption, and key generation by certificates is also utilized, making this work a hybrid IBE-PKI based approach. Also, in this work, the key management is handled by service delegation, where the service may prove a user's identity without online consent by the user, which adversely affects the privacy of the user. In [10], the authors present a multiple-authority ABE solution where the users may be revoked centrally. The scheme is secure up to $N - 2$ corrupt Trusted Authorities (TA) colluding, and the users can be revoked centrally. Similarly [11] suffers from a centralized and hierarchical approach for key management.

In [7], the users generate their public elements and compute their corresponding private key using a secret. The scheme proposed in [7] is inefficient compared to ours since it uses the ID-based mechanism to encrypt all the files stored in the cloud,

which is too costly for use in real-life applications. Additionally, [7] is based on the BF-IBE scheme [21], which uses secure hashing to group elements, which is inefficient. The key management is only user-centric; user files can not be recovered if a user can not access her key locally.

In [7], the authors propose a solution based on IBE. Their cryptographic scheme is based on Type 1 bilinear pairings. This type of pairings are inefficient and have less security level than Type-3 pairings. Specifically, it is not possible to construct a 192-bit and higher security level with a Type-1 pairing. Also, [12] is based on a single centralized PKG, which has the single-point-of-failure and presents a malicious CSP risk.

Bentajer et al. present CS-IBE [8], which is based on the original BF-IBE scheme [21]. However, this work does not present any cryptographic scheme or security analysis. It is based on the original Type-1 pairing IBE which has been shown to have security weaknesses [25]. Additionally, it is based on a single, centralized key management authority that constitutes a single point of failure and may result in a privacy breach in the case of a curious CSP. Similarly, Tan et al. [30] present a non-pairing based IBE scheme; however, their study also does not support multiple PKGs. In contrast, SCSS-SAKKE-IBE utilizes the fastest known pairing-based IBE scheme in literature, namely, SAKKE-IBE, and adopts the Type-3 pairings with SAKKE-IBE following [27]. Additionally, SCSS supports multiple, distributed PKGs, by adopting the scheme from [26].

2.2. Architecture and methodology

SCSS-SAKKE-IBE is based on two fundamental concepts: an Encryption-as-a-Service (EaaS) based architecture, and an IBC scheme. To the best of our knowledge, no EaaS solution based on Type-3 IBE exists in the literature.

In EaaS, encryption can be provided to the users without the need to implement any encryption application on their device, nor it uses any proxy similar to that in [31]. EaaS helps minimize the difficulty of encryption and key management tasks while maintaining data confidentiality. Figure 3 depicts the EaaS scheme adopted by SCSS-SAKKE-IBE, which makes encryption simpler, where the customer buys a pay-as-you-go service for secure file storage from their cloud security service provider. Particularly in IoT/IoMT (Internet of Medical Things) scenarios that store sensor data on the cloud, the devices lack the computational power to efficiently execute asymmetric cryptographic operations. In healthcare systems, IoMT devices require the high level of security provided by asymmetric cryptography. In addition, the IoMT devices are heterogeneous, which makes key management for these devices highly complex. The key server and the encryption server may be placed under complete control of the cloud users or their organization, which protects security mechanisms as well as user data from malicious or curious CSP. By offloading the key management operations to the key server the IoMT devices benefit from high level of security with central manageability. The SCSS-SAKKE-IBE system prevents the possibility of the EaaS provider to access clear-text user data. In IBE, the encryption is done over the public key of the user while the decryption is done over the private key of the user. Assuming the user keeps

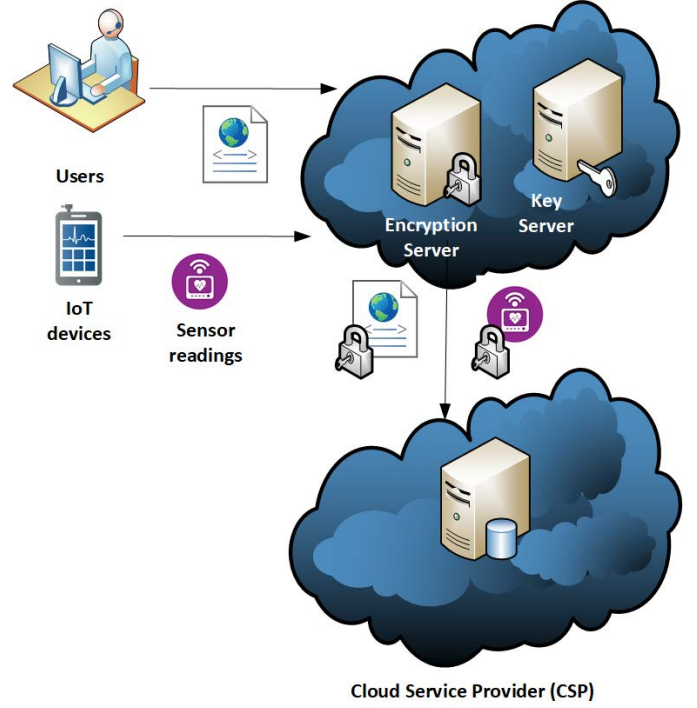


Figure 3: Encryption as a Service (EaaS) concept

his/her private key safe, which is stored in a secure location in the users' responsibility, the EaaS never will be able to decrypt user data.

The key generation method in IBE introduces a key escrow mechanism which requires a trust in the EaaS service provider. For overcoming this problem, a multiple PKG method is used, based on the Identity Based Non-Interactive Key Exchange (IB-NIKE) [32, 33], for the distribution of trust. An additional Trusted Third Party (TTP) has been utilized to sign the encrypted files, to enhance the security of the system and distribute the trust at the Encryption Service (ES). In this regard, no individual party, including the EaaS provider, can have access to a key for obtaining clear-text data for a user. Furthermore, we have included an additional level of security to protect user accounts and data. A user account is locked with a common agreed key by three trusted parties, ES, Trusted Key Generation Authority (TKGA), and Legal Authority (LA). This prevents the digital forensic team to access user files during normal system operation. When a legal request for forensic investigation is received by the TKGA that hosts the KMS, the common shared key is regenerated with the involvement of three trusted parties. The user account can be unlocked for forensic investigation only based on the agreement of three trusted parties.

The authors of [34] identified the most important challenges in IoT forensics as follows: identification, collection, preservation, analysis and correlation, attack attribution, and evidence presentation. In addition, encryption is identified in their study and the study of Arshad et al. [35] as an important challenge that and presents a 'trade-off' between privacy of the user and the success of the forensics investigation. In our study, we present a solution for this tradeoff, where the user privacy can be pro-

ected in the cloud through strong ECC-based encryption, at the same time, allowing the authorities to conduct digital forensics on encrypted data, through a multiple PKG based key escrow mechanism.

The main components of SCSS-SAKKE-IBE are the User Client (UC), Encryption Service (ES), Key Management Service (KMS), and Storage Service (SS). The main components and their relations are depicted in Figure 4.

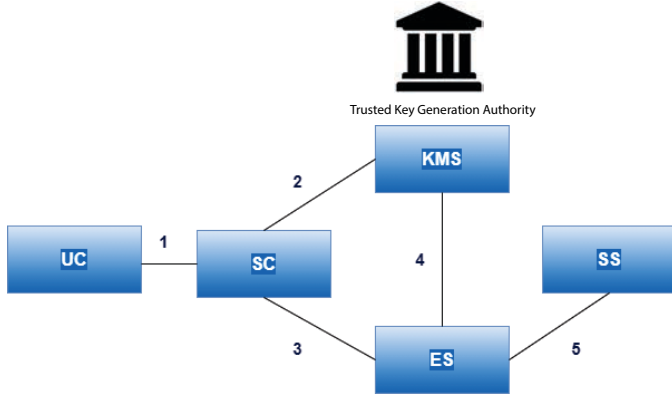


Figure 4: Main components of SCSS-SAKKE-IBE

The main actors for secure cloud storage are the end-users and the CSP. The end users are the data owners whose data need to be protected in the cloud. CSPs store encrypted user data by the SS and have no further control over the user data once it is encrypted by the SCSS-SAKKE-IBE. The main components of the SCSS-SAKKE-IBE and their functions are as follows:

1. User Client (UC). End-users access the system through the provided client application. A prototype UC is implemented as a secure browser-based application, which supports the required functionalities of security services. The UC is capable of running ID-based key encryption and decryption, symmetric file encryption and decryption, document signing, and signature verification in a sandboxed environment, where it is possible to run a native code that can access computation resources.

2. Service Controller (SC). All services listed here, client-service interactions, request and response management, and inter-service communications are controlled by SC. The SC also implements orchestration functionalities for other security services. SC may interact with external Identity Provider(s) (IdP) for cloud authentication; however, this interaction is out of the scope of this manuscript.

3. Key Management Service (KMS). KMS acts as a public key generator for SCSS-SAKKE-IBE to enable users and services to employ IBC requirements. It generates public parameters, the system's main secret key, and the public and private keys of each user for secure communication. The private keys of users are stored in a Hardware Security Module (HSM) for protection against tampering. The public key of a user is a string array formed by the identity of the user followed by a validity period. Each user has two private keys, one for encryption and the other for signing.

4. Encryption Service (ES). All encryption processes are the responsibility of the encryption service. Main functions include:

- Symmetric encryption and decryption
- ID-based key encryption and decryption
- ID-based signing and verification

5. Storage Service (SS). Storage of files to cloud storage services or local storage is done by this service. In addition to using a local file system, a user has the option to integrate external cloud storage options such as Dropbox, Box, Google Drive.

The high-level description for the steps of a user cloud secure storage use case, illustrated in Figure 5, is described as follows. In Figure 5 the services described below are simplified within a single actor named SCSS.

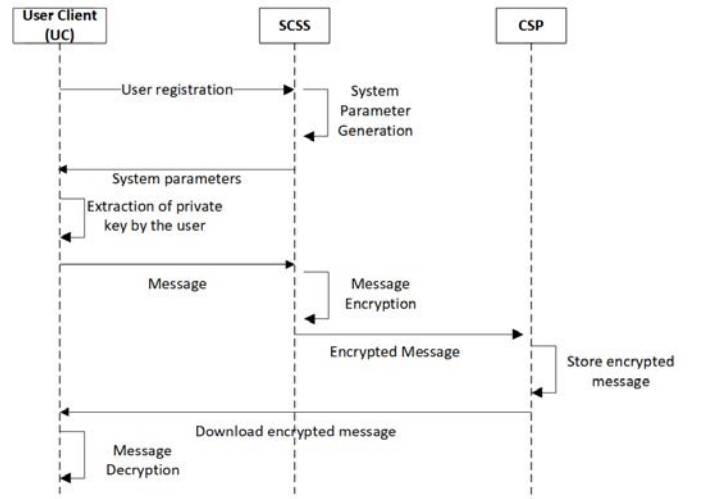


Figure 5: Main secure cloud storage scenario

Listing 1: Main flow of secure file storage on the cloud

1. A user logs into SCSS using UC with his ID and password.
2. UC sends this login request to it to SC.
3. SC checks the login request and redirects it to IdP.
4. IdP checks the user credentials, permits user to login sending UC acknowledgment and user's id-based private key.
5. UC then requests the list of folders from SC which redirects this to SS.
6. SS gets the user's stored files from CSPs and sends this information to UC through SC.
7. UC lists all these files to the user.
8. User then chooses the file that she wants to encrypt either from the list or from her computer.
9. UC signs the file with its private key, creates a file key, encrypts the file with this key, encrypts the key with ID of SCSS and sends this file package to SC which redirects this to ES.

10. ES decrypts the key, decrypts the file with this key and verifies the signature on the file.
11. ES then encrypts this file with an encryption key obtained from KMS.
12. ES then sends the encrypted file to SS which sends it to the chosen CSP and this completes the scenario.

2.3. Use case for forensic investigations on encrypted data in the cloud

The operation of this use case is depicted in Figure 6.

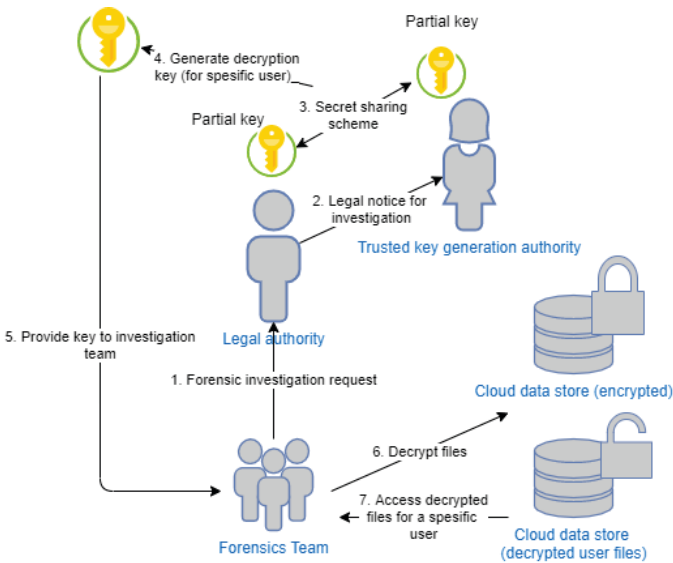


Figure 6: Digital forensic investigation on encrypted data on the cloud

The steps in Figure 6 are as follows:

Listing 2: Main flow of forensics investigation on encrypted data

1. Forensics team sends a forensics investigation request to the legal authority. In this request they will specify a user ID for which the investigation will take place.
2. The legal authority evaluates this request, and upon approval sends a legal notice for investigation to the trusted key generation authority.
3. The key generation authority and the legal authority regenerate the private key for the user with their partial keys. Based on the private key the IdP generates decryption keys for the user. The private key is then destroyed to prevent its further use.
4. The IdP provides decryption keys to the investigation team.
5. The forensics team requests the decryption of the files for the investigation from the ES.
6. The forensics team accesses the decrypted files and conducts the investigation.

7. After the forensic investigation, the user account is locked through the three-party key agreement protocol defined in Section 3.6. When the user account is locked, only the user can access the files in the account.

2.4. Use case for user management and identity federation for forensic team

Privileged user management provides the special requirements to manage the life-cycle of user accounts with higher privileges in a system, such as the forensic team. This also fulfills requirements to authenticate, authorize, log, monitor and audit the access of privileged users. In SCSS-SAKKE-IBE the forensic team users can authenticate to SCSS-SAKKE-IBE through an external ID provider with identity federation. The access is logged for further auditing. The use case steps for forensic team user authentication is given in Listing 2.4

Listing 3: Identity federation for authentication of forensics team with external IdP

1. Forensic team user sends an access request to SCSS.
2. SCSS redirects forensic team user to identity provider with authentication request.
3. Identity provider processes the authentication request.
4. Forensic team User is redirected to IDP authentication URL.
5. Forensic team User authenticates at IDP.
6. Identity provider generates Single Sign-on Assertion with persistent name identifier from user account.
7. Forensic team User is redirected back to SCSS using POST with the Single Sign-on Assertion.
8. SCSS validates assertion, and finds the local account based on the persistent name identifier.
9. SCSS creates Single Sign-on token for the local account.
10. SCSS records the system access for further auditing.

3. Cryptographic Scheme

This section describes the SCSS-SAKKE-IBE cryptographic scheme, which is based on the IEEE 1363.3 standard [36] and the SAKKE-IBE protocol [24] with Type-3 [25, 27], and Multiple PKG extensions [26]. The scheme includes system parameter generation, message encryption, decryption and key establishment for secure communication based on IBC.

In IBE, while public keys are generated from identifications of users, private keys corresponding to public keys are obtained by a special entity called Private Key Generator (PKG). PKG is a trusted third party that generates users' private keys. For this

purpose, PKG has to generate its master public key pk_{PKG} and master private key sk_{PKG} and has to make pk_{PKG} available to his services. In SCSS-SAKKE-IBE, the KMS acts as a PKG.

Let $(G_1, +)$ and $(G_2, +)$ be two additive cyclic groups of order q with $G_1 = \langle Q \rangle$ and $G_2 = \langle P \rangle$, (G_3, \cdot) be a multiplicative cyclic group of order q , where q is a prime number. 0 denotes the identity elements of the groups G_1 , G_2 and 1 by the identity element of G_3 . Assume that the Discrete Logarithm Problem (DLP) is hard in both G_1 and G_2 (i.e., given a random $R \in G_1$ (or $\in G_2$), it is computationally infeasible to find an integer $x \in \mathbb{Z}$ such that $R = x \cdot Q$). A pairing is a map $e: G_1 \times G_2 \rightarrow G_3$ satisfying the following properties [23]:

Bilinearity: For all $P_1, Q_1 \in G_1$ and $P'_1, Q'_1 \in G_2$, e is a group homomorphism in each component, i.e.

$$e(P_1 + Q_1, P'_1) = e(P_1, P'_1) \cdot e(Q_1, P'_1) \quad (1)$$

$$e(P_1, P'_1 + Q'_1) = e(P_1, P'_1) \cdot e(P_1, Q'_1) \quad (2)$$

Non-degeneracy: e is non-degenerate in each component, i.e., For all $P \in G_1$, $P \neq 0$, there is an element $Q \in G_2$ such that $e(P, Q) \neq 1$. For all $Q \in G_2$, $Q \neq 0$, there is an element $P \in G_1$ such that $e(P, Q) \neq 1$.

Computability: There exists an algorithm which computes the bilinear map e efficiently.

There are many pairing related hard problems. The basic hardness assumption of the cryptosystems which use bilinear maps is to compute the inverse of the function e and the DLP is intractable in the groups G_1 , G_2 and G_3 . Note that bilinear maps are generally obtained from Weil or Tate pairing of elliptic curves in practice [37]. Due to numerous security and efficiency tradeoffs, the Type-3 bilinear maps were used in light of [25] and [27] in the design of our IBC based mechanism.

In the description of the protocol which is described in the following sub-sections, $H_1: \{0, 1\}^* \rightarrow \mathbb{Z}_q^*$ is a hash function as per the IEEE 1363.3 standard [37] and $H_2: \{0, 1\}^* \rightarrow \mathbb{Z}_n$: n represents the message space length in bits, where $(n \approx \log_2(q) + 1)$.

3.1. System Parameter Generation

System parameters are generated according to the IEEE 1363.3 standard [36] and SAKKE-IBE protocol [26] as follows: Inputs are integers: ℓ_1, ℓ_2, p, q . Outputs are $(SystemPublicKey, SystemMasterSecretKey) = ((p, q, e, \mathbb{G}_T, \mathbb{G}_1, \mathbb{G}_2, H_1, H_2, P_1, P_2, g, Q), s)$, where: p and q are prime numbers, $\mathbb{G}_T, \mathbb{G}_1, \mathbb{G}_2$ are groups, P_1 and P_2 are generators, g is a group generator calculated by use of the matching function e ; which is chosen accordingly, s is the master secret key of the system; which is a random number generated based on ℓ_2 and $Q = s \cdot P_1$ is the public key of the system. The private key of the encryption service is calculated as follows:

$$priv = (s + H_1(ID))^{-1} \cdot P_1 \quad (3)$$

where ID is the identity of the entity for which a private key is to be generated.

3.2. Distribution of Trust with Multiple PKGs

Since the PKG is a trusted entity, using a single PKG may compromise the privacy of user information. The addition of multiple PKG support eliminates the user to trust a single PKG. The trust distribution mechanism in SCSS-SAKKE-IBE extends the works of Kate et al. [26] and Sakai, Ohgishi and Kasahara [32] by supporting Type-3 IBE schemes. The key exchange in the system is performed through the Identity Based Non-Interactive Key Exchange (IB-NIKE) scheme has been originally proposed in [32]. Chen et. al. [33] proposed an adaptively secure IB-NIKE scheme in the standard model that does not explicitly require multilinear maps.

An IB-NIKE scheme consists of the following polynomial-time algorithms:

- *Setup*(κ, n): on input a security parameter κ and a parameter n for the number of participants, output master public key mpk and master secret key msk . Let I be the identity space and SHK be the shared key space.
- *Extract*(msk, id): on input msk and an identity $id \in I$, output a secret key $skid$ for id .
- *Share*($skid, I$): on input secret key $skid$ for identity id and an list $I \in I^n$ consisting of n identities, output a shared key shk for I . We assume that the identities in I are always lexicographically ordered.

In the multiple PKG case, the distributed PKG setup takes place as follows: When generating user keys with multiple PKGs, each PKG generates its master-secret share $s_i \in \mathbb{Z}_p$ and the public key tuple $C_{(g)}^{(s)} = [g^s, g^{s_1}, \dots, g^{s_n}]$. The master-secret shares are used to generate private key shares users for them to extract their private keys.

The key extraction mechanism is utilized in the presence of distributed PKG defined in [26]. The users extract their private keys with information received from multiple PKGs as follows:

Input: User ID, public key tuple $C_{(g)}^{(s)}$, share of user private key d_i^{ID} Output: User private key d^{ID} Private key extraction:

1. The user with identity ID contacts every node, every honest node P_i authenticates the user client.
2. The nodes return their share of the user's private key d_i^{ID} .
3. The user verifies the shares of user private key d_i^{ID} .

The user extracts his/her private key d^{ID} using interpolation.

The users share their public keys with the *Share* function as follows.

Input: $sk_{id_a}, I = (ida, idb), I = (idb, ida)$ Output Shared key $shk = G(e(sk_{id_a}, H(id_b)))$.

3.3. Message Encryption

The message encryption function inputs the message m , system public key Q , and the identity of the recipient ID and generates an encrypted text E . The operation of the message encryption function is as follows. Here, r and R are randomization

parameters and S is a value calculated for masking the message.

$$id = H_1(ID) \quad (4)_{635}$$

$$r = H_1(m \parallel id) \quad (5)$$

$$R = r \cdot (id \cdot P_1 + Q) \quad (6)$$

$$S = m \oplus H_2 g^r \quad (7)$$

$$E = (R, S) \quad (8)$$

3.4. Message decryption

The message decryption function inputs the identity of the sender ID, encrypted text E , private key, system public key Q and outputs a clear text message m .

$$id = H_1(ID) \quad (9)$$

$$(R, S) = E \quad (10)_{640}$$

$$w = e(R, priv) \quad (11)$$

$$M = S \oplus H_2(w) \quad (12)$$

$$r = H_1(m \parallel id) \quad (13)$$

The message is validated message by checking $r \cdot (id \cdot P_1 + Q) = R$. If message is validated, the function outputs $m = M$.

3.5. Key Establishment for Secure Communication

As the first layer of defense, the Transport Layer Security (TLS) protocol is utilized for all communication between services. An additional secure communication layer is proposed. The reason for the additional security is that there are many recent attacks on the TLS protocol [38, 39, 40] and the security of the protocol is highly implementation dependent. An identity based key establishment mechanism is utilized for additional security. This mechanism is utilized both for the trusted connection of users to the SCSS-SAKKE-IBE services and for secure communication between the SCSS-SAKKE-IBE services. This mechanism is based on the SCK-2 [41] key agreement protocol. Others like [42] used bilinear Diffie-Hellman (BDH) approach for the IoT scope. The hash functions utilized for key establishment are defined as follows:

$$H_3 : \{0, 1\}^* \rightarrow \mathbb{G}_2 \quad (14)_{655}$$

$$H_4 : \{0, 1\}^* \times \mathbb{G}_2 \times \mathbb{G}_2 \times \mathbb{G}_2 \times \mathbb{G}_T \times \{0, 1\}^* \rightarrow \{0, 1\}^n \quad (15)$$

The inputs to the key establishment function, id_A, d_A, id_B, d_B, Q are calculated as described below. id_A , public key and d_A , private key for message source A , which is a user or a service are calculated as follows:

$$id_A = H_3(ID_A) \quad (16)$$

$$d_A = s \cdot id_A \quad (17)$$

id_B , a public key and d_B , a private key for message destination B , which is a service are calculated as follows:

$$id_B = H_3(ID_B) \quad (18)$$

$$d_B = s \cdot id_B \quad (19)$$

The master public key of SCSS, Q is calculated as follows:

$$Q = s \cdot P_1 \quad (20)$$

Key establishment between A and B takes place as follows:

1. A chooses random $x \in \mathbb{Z}_q$ and sends $E_A = x \cdot P_1$ to B .
2. B chooses random $y \in \mathbb{Z}_q$ and sends $E_B = y \cdot P_1$ to A .
3. A and B calculate K_1 and K_2 which are used to establish a common key as follows. A calculates K_1 and K_2 :

$$K_1 = e(Q, x \cdot id_B) \cdot e(E_B, d_A) \quad (21)$$

$$K_2 = x \cdot E_B = xy \cdot P_1 \quad (22)$$

On the other side of the communication, B also calculates K_1 and K_2 :

$$K_1 = e(Q, y \cdot id_A) \cdot e(E_A, d_B) \quad (23)$$

$$K_2 = y \cdot E_A = xy \cdot P_1 \quad (24)$$

And finally the common session key between A and B , SK is calculated by both sides A , and B as follows:

$$SK = H_4(A, B, E_A, E_B, K_1, K_2) \quad (25)$$

3.6. Three-party Key Agreement Protocol for User Account Protection

As an additional level of security, we introduce the concept of locking a user account with a common key agreed by three trusted parties. In SCSS-SAKKE-IBE, the ES, Trusted Key Generation Entity, and Legal Entity are the three trusted parties who can lock a user account. This additional security mechanism prevents a digital forensic team to access user files during normal operation. When a user account is unlocked then the digital forensic team can access the user account and decrypt files for forensic investigation. The three-party key agreement protocol is based on Joux et. al. [43] as follows:

Where parties A, B, C possess secret keys $a, b, c \in \mathbb{Z}_q^n$;

- A sends aP_1 to B, C
- B sends bP_1 to A, C
- C sends cP_1 to B, A
- Key computed by A is $K_A = e(bP_1, cP_1)^a$
- Key computed by B is $K_B = e(aP_1, cP_1)^b$
- Key computed by C is $K_C = e(aP_1, bP_1)^c$

The common agreed key K_{ABC} is as follows: $K_{ABC} = K_A = K_B = K_C = e(P_1, P_1)^{abc}$. The basic Joux scheme is known to be vulnerable to man-in-the-middle attacks. In order to prevent this attack, we provide the authentication for communication such that the three-party key agreement protocol takes place over the secure communication protocol explained in Section 3.5. The locking key is used once and after unlocking for forensic investigation it has to be regenerated. This mechanism prevents the need for regenerating the user keys.

4. Results and Discussion

4.1. Security and Computational Time Analysis

This section analyses the security, ease of use and management, performance and efficiency of the proposed SCSS-SAKKE-IBE cryptographic scheme.

4.1.1. Security Analysis

For the security analysis, we assume that the services provided by the SCSS-SAKKE-IBE are honest but curious, but the CSP may be malicious. In the IBE security notions proposed in [29] the adversary has access to an extraction oracle, that inputs an identity $id_ch \in 0, 1^*$ and outputs the corresponding private key. The adversary has a challenge, which consists of a pair (id_ch, c) , where c is a ciphertext produced by encryption under the identity id_ch . The adversary is allowed to adaptively select id_ch , probably depending on the information received so far (such as decryption keys). This adversary is deemed to be an IND-ID-CCA adversary. An identity-based encryption scheme \mathcal{E}_{ID} is considered to be IND-ID-CCA secure if for any IND-ID-CCA adversary, for an attacker under a selected identity id_ch , the probability of guessing a random encrypted bit b within the ciphertext c is negligible.

The Sakai-Kasahara (SAKKE) Type 3 pairing scheme has been proven secure in the random oracle model against chosen-ciphertext attacks [27]. According to this study, the proposed scheme is secure up to IND-ID-CCA. Additionally according to the security proof presented in [26] the multiple-PKG extension of the SAKKE IBE is secure up to IND-ID-CCA. The security of SK-IBE with a distributed (n, t) PKG is based on the BDHI (bilinear Diffie-Hellman inversion) assumption [26].

Table 2: Comparison of computational time

Operation	ABE [19][20]	BF-IBE [16][10][7]	SCSS-SAKKE-IBE
Public key gen	1	m	M
Private key gen	1	$h + m$	$h + m$
Encrypt	$3e$	$> 2p + m$	$2p + m$
Decrypt	$(l + 2)p$	$> e$	E

4.1.2. Ease of Use and Management

IBC avoids trust problems encountered in certificate based PKIs. In PKI, the key pair is generated from random information which results in a requirement for a certificate to bind the public key to its particular use. In contrast, the key pair in an IBC environment is generated explicitly from publicly available information. This information could be the identity of the user in the system. As a result, a client A can generate the public key of another client B without the need for a directory search or a public key transfer from B to A . Another difference is that in IBC, user private keys are generated based on a master secret which is known to a PKG. In contrast, in a PKI-based solution, the CA is concerned with generating a certificate without a requirement to generate private keys by the CA. Despite this may seem like a useful functionality, it introduces the need of CRL, which become complex and costly to manage for many enterprises.

The main barriers to cloud acceptance for forensics experts are privacy and data security because the cloud system users and providers are not in the same trusted domain. The diverse technologies and algorithms are used to preserve and control security and privacy in the Cloud. These methods primarily involve encryption, authorization and access control.

The essential method is PKI based on the PKI algorithms. In traditional PKI, cloud users encrypt their data with their public key before sending it to the cloud systems. When a cloud user requires access to his/her data, the cloud system gives the private key for the encrypted data. The cloud user decrypts this encrypted data in the cloud with the private key. This method of operation reduces the privacy of the user because the CSP is able to access the user files through the stored private key.

On the other hand, access to the data may be required for forensic investigations. Encrypted storage of data on the cloud is an obstacle for forensic teams. In particular, the e-mail and messaging applications' encrypted content creates a problem in terms of accessing evidence. In this study, the user data on the cloud can be accessed by re-obtaining the encryption key based on the consent of all trusted authorities. However, during normal operation, it is not possible to access user data by any authority except the user, thanks to a three-key handshake mechanism for user account protection. This adds to the ease of use and management of the system as well as increased security.

4.1.3. Complexity and Efficiency Analysis

The Sakai-Kasahara (SAKKE) scheme utilized in this work is the most efficient IBE scheme in literature. It does not require a pairing operation for encryption and only requires a single pairing computation for decryption. There is an extra group exponentiation operation, however, this is far less costly than

Table 3: Comparison of bandwidth requirements

Operation	PKI-based	IBE (BF-IBE)	SCSS-SAKKE-IBE
Operation Cipher-text size	$ PK + m $	$ G1 + n + m $	$ m $
Public key verify	$ PK $	0	0
CRL download	100 – 500K	0	0

805

pairing computations. There is no need in SAKKE to calculate a hash into an elliptic curve group, which is costly and also poses security risks if the groups are not chosen carefully. Similarly, since the lack of necessity of a pairing computation on the operation and the need to hash into an elliptic curve group, performance is improved against other IBE based schemes. Computational time analysis is provided in Table 2.

In Table 2, p denotes a pairing, m a scalar or modular multiplication, h a hashing function and e an exponentiation operation. The costs for these operations are ranked as: $C(p) > C(e) > C(h) > C(m)$. For ABE, l denotes the number of attributes. Although ABE provides better performance for key generation, the ABE decryption cost is much higher than BF-IBE and SAKKE and this is undesirable since in real-life scenarios, decryption operations occur much more frequently than key generation operations. Additionally, decryption operations in the cloud storage occur more frequently than encryption operations, since a file is mostly uploaded once and accessed many times.

Analysis results of the bandwidth requirements of our proposal against other approaches is given in Table 3. In Table 3, $|m|$ denotes the message size, $|PK|$ denotes the public key size of RSA (generally 2048 bits), $|G1|$ denotes the number of bits necessary to represent group $G1$ (depends on the choice of the group), n is the elliptic curve size (Typically 160 bits). From Table 3, it is evident that the bandwidth requirements of SCSS-SAKKE-IBE are superior to both other IBE schemes and the PKI-based approach. Since IBE for key encryption is used and the message encryption takes place by derived symmetric keys, the cipher-text size is simply equal to the message size and does not suffer from the overhead incurred by other IBE schemes. Our proposal minimizes the key distribution complexity in comparison to the PKI-based approach, which requires CAs and longer keys. The communication overhead is reduced since the public key verification operations and revocation controls can be made offline. CRL downloads are not necessary for SCSS-SAKKE-IBE since IBE schemes do not require certificates.

4.2. Experimental results

This section compares the cryptographic scheme in SCSS (SCSS-SAKKE-IBE) to various other proposed cryptographic schemes in the literature. Our aim is to compare the Type-3 pairing performance of SCSS-SAKKE-IBE to other Type-1 pairing based systems as well as IBE extensions such as HIBE and ABE. The comparisons will be based on the security levels comprising 80 bits, 128 bits, 192 bits, and 256 bits. The implementations and benchmarks are based on the MIRACL cryptographic library [44]. The benchmarks have been conducted on a computer with Intel Core i7-8550U CPU, 16 GB RAM and

256 GB SSD hard drive. The main comparison parameters will be as follows:

- Setup time: This is the time taken to generate the system parameters. This is generally done once at the beginning of the system operation.
- Private key generation time: This is the time taken to generate user private keys. This is generally done at the time of user registration.
- Encryption time: This is the time taken to generate session keys for encryption of messages as well as the encryption of the message itself.
- Decryption time: This is the time taken to generate the session keys for decryption at the receiver as well as the decryption of the message itself.

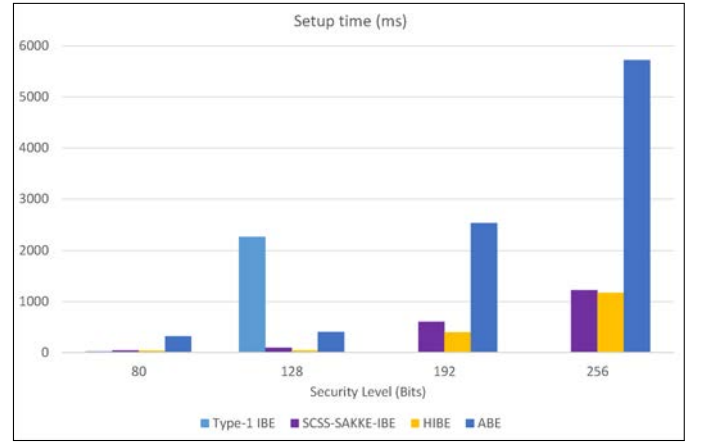


Figure 7: Setup time

The purpose of the first experiment is to benchmark the System parameter generation step in the proposed cryptographic scheme, which comprises of the measurement of the setup time parameter. The purpose of the second experiment is to benchmark the extraction of the private key by a user step in the proposed cryptographic scheme, which comprises of the measurement of the private key generation time. The purpose of the third and the fourth experiments is to benchmark the message encryption and message decryption steps in SCSS-SAKKE-IBE, which comprise of the measurement of encryption and decryption times, respectively.

In Figure 7, the performance is related to the presented setup time. According to benchmarks it takes around 1s to generate the system parameters in SCSS-SAKKE-IBE. This is the time to generate the system parameters for a single PKG, therefore in the multiple PKG case, the setup time will be linear with the number of PKGs. It should be noted that the setup time in SCSS-SAKKE-IBE is about 6 times less than ABE schemes and 3 times less than Type 1-based IBE schemes. The setup time is slightly higher than the HIBE scheme (around 10%). Since this operation takes place only once, the total time delay

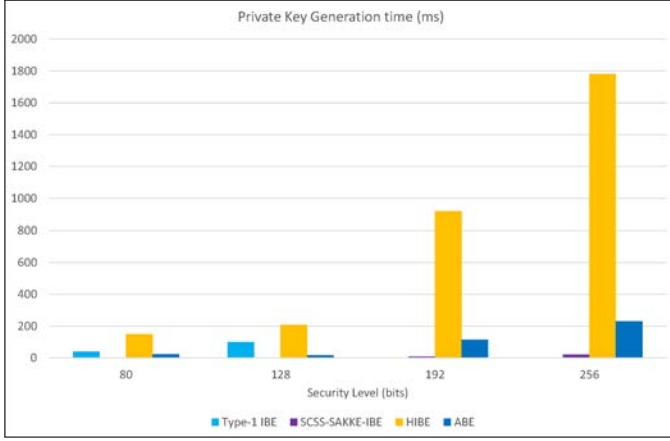


Figure 8: Private Key Generation Time

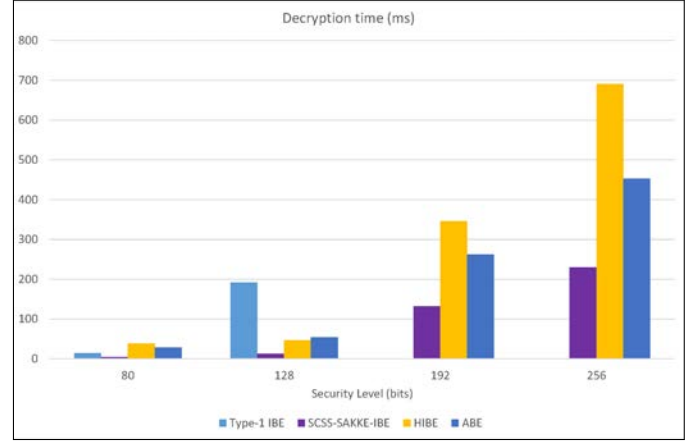


Figure 10: Decryption Time

to setup the system in a multiple PKG scenario with SCSS-SAKKE-IBE in comparison to HIBE would be negligible (for 5 PKGs, a total of 0.5s difference).

For the entities to be able to communicate in the system their public-private keys need to be generated. Figure 8 shows the key generation time is linear with the security level, while it is exponential in the other compared schemes. For Type-1 based IBE schemes there are no results for 192 and 256 bit security level since Type-1 schemes only provide security up to 128-bits. For the 128-bit security level which is comparable to the Type-1 based IBE, the proposed scheme exhibits up to 100 times performance increase, while it also performs about 200 times better than HIBE. As per ABE, the proposed scheme presents about 18 times performance increase. For the 256-bit security level, we compare the proposed scheme with HIBE and ABE. Here, a private key generation in SCSS-SAKKE-IBE takes around 20ms while HIBE takes around 1800ms and ABE takes around 230ms. If we consider that a private-public key pair has to be generated for each user, for large systems, HIBE and ABE based schemes present a scalability bottleneck. Generating keys for 1000 users in HIBE would take around 30 minutes, and in ABE around 4 minutes. Whereas in the proposed SCSS-SAKKE-IBE this operation would be completed in 30s.

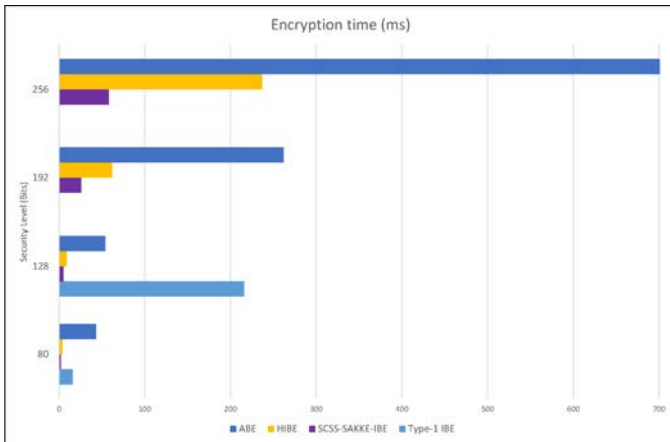


Figure 9: Encryption Time

The encryption operation takes place whenever a file will be encrypted and uploaded to the cloud. In the proposed scheme, the encryption keys are generated once for a file using the public key of the user, and this key is then used for symmetric encryption with AES. Since the symmetric encryption is linear and the same for all schemes, it has not been included in the time delay benchmark of Figure 9. The difference between various IBE-based schemes on the encryption side is based on the time cost of generating the symmetric key for encryption. For this operation, the ABE-based schemes perform the worst, followed by HIBE. For any security level higher than 128 bits, SCSS-SAKKE-IBE outperforms existing schemes, i.e. HIBE scheme by 4 times, and ABE scheme by 12 times. The encryption operation is a frequently used operation and will take place each time a file is encrypted to be uploaded to the cloud. Therefore, this performance improvement will make a very significant difference for the end user.

The decryption operation takes place whenever an encrypted file is downloaded from the cloud and decrypted. This operation is expected to occur more frequently than the encryption operation, therefore, it has more impact on the overall performance. Here SCSS-SAKKE-IBE provides a significant performance difference with the other schemes, for example, in the 128-bit security, SCSS-SAKKE-IBE outperforms HIBE by 3 times, ABE by 4 times and Type-1 IBE by 15 times. For higher security levels there is no Type-1 equivalent as described before, and against HIBE, the performance improvement is 3 times whereas against ABE the performance improvement is 2 times. It takes on average 230ms for the decryption operation in SCSS-SAKKE-IBE for the 256-bit security, whereas it takes HIBE around 700ms and ABE around 450ms for the same decryption operation. The performance charts for the decryption operation are provided in Figure 10.

We now provide results demonstrating the scalability of SCSS-SAKKE-IBE. For the Type-1 IBE, security levels higher than 128-bits are not applicable. For the comparison of scalability including Type-1 IBE, first we provide the encryption and decryption scalability results for the 128-bit security level. The horizontal axis varies the number of users from 1 to 10000. In Figure 11, the vertical axis shows the total time taken for

encryption for the Type-1 IBE, SCSS-SAKKE-IBE, HIBE and ABE schemes. In Figure 11, the vertical axis shows the total time taken for decryption for the Type-1 IBE, Type-3 SAKKE-IBE, HIBE and ABE schemes. As seen from Figures 11 and 12, SCSS-SAKKE-IBE and HIBE schemes demonstrate the highest scalability of encryption for the 128-bit level. However, when it comes to decryption, SCSS-SAKKE-IBE has better scalability than all of the other compared schemes.

Then, we provide the experimental results demonstrating the scalability of the proposed scheme for the 256-bit security level. For this security level, the SCSS-SAKKE-IBE, HIBE and ABE schemes are compared, because Type-1 IBE does not support this security level. For the 256-bit security level, the scalability of the proposed scheme is considerably higher than the other schemes. The decryption operation takes place whenever a file is read from the cloud, therefore, the superior scalability with respect to the decryption operation is an important advantage of the proposed scheme.

5. Conclusion

This paper proposed a secure cloud storage system, named SCSS, for a cloud-based storage environment without the need for heavyweight certificate management, with benefits in terms of protection, scalability, and efficiency. The underlying cryptographic scheme (SCSS-SAKKE-IBE) is highlighted to outperform other pairing based IBE-based schemes in the literature. Furthermore, the proposed scheme is shown to be IND-ID-CCA secure in the random oracle model.

The main contributions of the proposed work can be summarized as follows: The proposed SCSS-SAKKE-IBE scheme is based on Type-3 pairings and is more secure and efficient than the existing Type-1 pairing-based IBE schemes. SCSS includes a multiple PKG key management scheme adopted for Type-3 SAKKE-IBE scheme, which is the first in literature. In this way SCSS supports for multiple PKGs. In this regard, SCSS provides the means to distribute the trust of key management into multiple authorities, which includes distributed key management and encryption mechanisms and supports for multi-

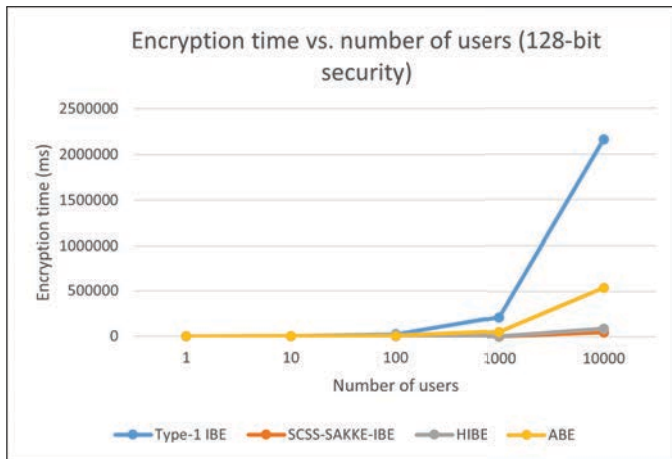


Figure 11: Encryption time vs. number of users in 128-bit security level

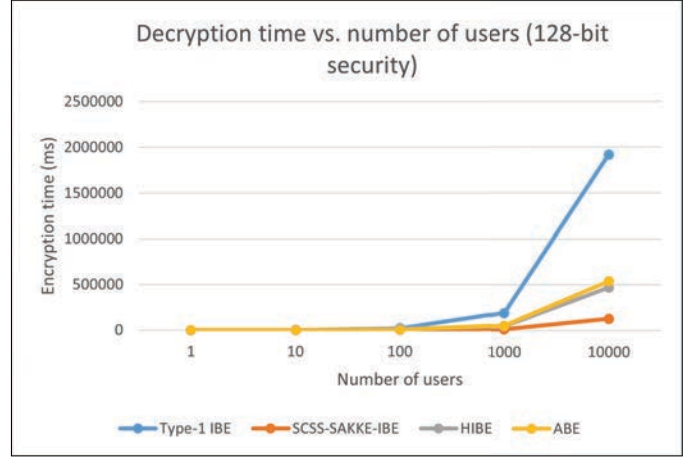


Figure 12: Decryption time vs. number of users in 128-bit security level

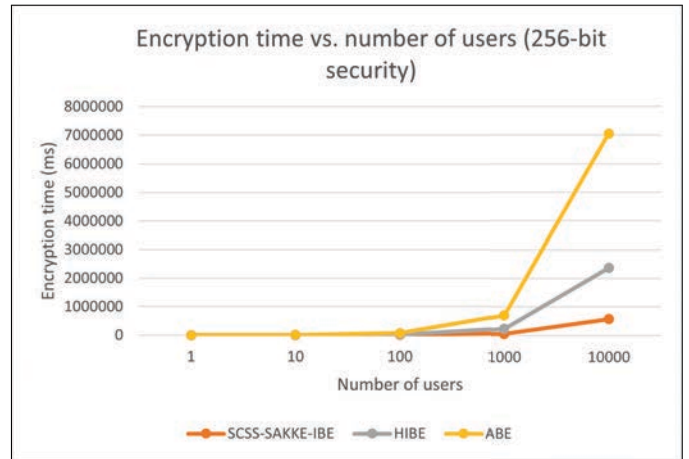


Figure 13: Encryption time vs. number of users in 256-bit security level

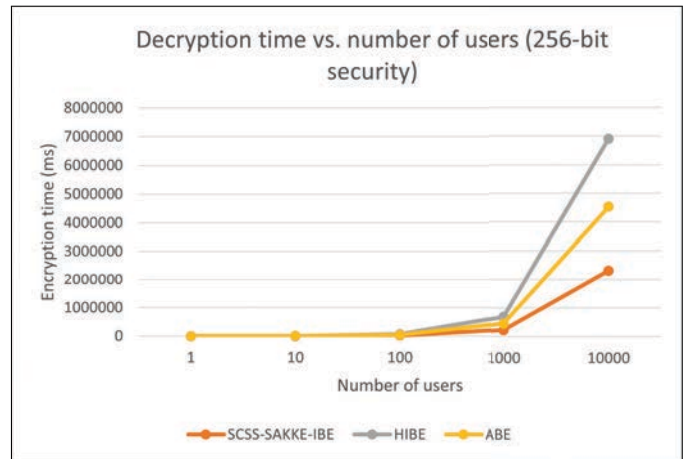


Figure 14: Decryption time vs. number of users in 256-bit security level

ple PKGs. During forensic investigations, the legal authorities will be able to use the multiple PKG mechanism for data access during investigations, whereas this mechanism also prevents a single authority to access user data by the distribution of the trust. Additionally, analytical and experimental results

prove that SCSS outperforms existing IBE based cloud security solutions. The SCSS-SAKKE-IBE scheme outperforms other pairing based IBE schemes. Especially, the SCSS-SAKKE-IBE scheme has very low private key generation, encryption and decryption times. The private key generation, encryption, and decryption and the Type-3 pairing scheme of SCSS based on SAKKE clearly outperforms other types of IBE schemes such as HIBE and ABE as well as Type-1 based IBE. In addition, SCSS-SAKKE-IBE demonstrates higher scalability than other pairing based IBE schemes, especially for the decryption operation. The improvement in scalability increases for higher security levels, e.g. 256-bits). Our experimental results show that the proposed scheme performs well in IoT-based environments where the number of users are expected to be large. Additionally, the digital forensic processes are faster thanks to higher performance and scalability for decryption operations.

Our proposal protects against a malicious CSP, unless all of the trusted parties collude. We evaluate that this is a very low possibility since other trusted parties are usually government or independent entities overseeing the security of the system. Meanwhile, it is desirable to have a legal process for accessing encrypted data for forensic analysis purposes through a defined process, which excludes the malicious EaaS case. Mitigation against a curious/malicious EaaS provider is provided through a user account protection mechanism. Additionally, the EaaS provider can not generate a user's private key to access user data, since the system supports the key generation only through the involvement of multiple PKGs.

One of our further work plans is to investigate the architecture where some of the SCSS services could be hosted by untrusted CSP as described in [45]. Another interesting future research avenue is to apply SCSS-SAKKE-IBE in IoMT scenarios where sensor data is stored on the cloud and the devices lack the computational power to efficiently execute asymmetric cryptographic operations. In healthcare systems, IoMT devices require the high level of security provided by asymmetric cryptography. As the number of IoMT devices increases, key management becomes highly complex. By offloading the key management operations to a server, the IoMT devices benefit from high level of security with central manageability. We also plan to explore new approaches linking secure data storage with biometrics.

Funding Information

This publication was made possible by NPRP grant NPRP 10-0125-170250 from the Qatar National Research Fund (a member of Qatar Foundation). The statements made herein are solely the responsibility of the authors.

Authors Biography



Devrim Unal is a Research Assistant Professor of Cyber Security at the KINDI Center for Computing Research, College of Engineering, Qatar University. He obtained his M.Sc. degree in Telematics from Sheffield University, UK and Ph.D. degree in Computer Engineering from Bogazici University, Turkey in 1998 and 2011, respectively. Dr. Unal's research interests include cyber-physical systems and IoT security, wireless security, artificial intelligence and blockchain.



Abdulla Khalid Al-Ali obtained his M.S. degree in Software Design Engineering and Ph.D. degree in Computer Engineering from Northeastern University in Boston, MA, USA in 2008 and 2014, respectively. He is an active researcher in cognitive radios for smart cities and vehicular ad-hoc networks (VANETs). Dr. Abdulla is currently head of the Computer Science and Engineering department at the College of Engineering in Qatar.



Ferhat Ozgur Catak received B.Sc. degree in electrical/electronic engineering in 2002 and his Ph.D degree in Informatics in 2014. Previously, he was chief researcher/associate professor of cyber security institute at TUBITAK in Turkey. Since 2020, he has been researcher with Simula Research Laboratory, Oslo, Norway. His research areas are cyber security, malware analysis, secure multiparty computation and privacy methods.



Mohammad Hammoudeh received his M.Sc. degree in Advanced Distributed Systems from the University of Leicester in 2006 and the Ph.D. degree in Computer Science from the University of Wolverhampton in 2008, both in the U.K. He is a Professor of Cyber Security in the Department of Computing and Mathematics at the Manchester Metropolitan University. Mohammad heads the CfACS Internet of Things Lab he founded in 2016. His research interests are in the areas of cybersecurity, the Internet of Things and complex highly decentralised systems.

References

- [1] Cloud storage market | growth, trends, and forecast (2020-2025), <https://www.mordorintelligence.com/industry-reports/cloud-storage-market> (Accessed on 03/04/2021) (2020).
- [2] K. Awuson-David, T. Al-Hadhrani, M. Alazab, N. Shah, A. Shalaginov, Bcfl logging: An approach to acquire and preserve admissible digital

- forensics evidence in cloud ecosystem, *Future Generation Computer Systems* 122 (2021) 1–13. doi:<https://doi.org/10.1016/j.future.2021.03.001>. URL <https://www.sciencedirect.com/science/article/pii/S0167739X21000807>
- [3] M. Masud, M. Alazab, K. Choudhary, G. S. Gaba, 3p-sake: Privacy-preserving and physically secured authenticated key establishment protocol for wireless industrial networks, *Computer Communications* 175 (2021) 82–90. doi:<https://doi.org/10.1016/j.comcom.2021.04.021>. URL <https://www.sciencedirect.com/science/article/pii/S0140366421001651>
- [4] S. Zawoad, R. Hasan, Trustworthy digital forensics in the cloud, *Computer* 49 (3) (2016) 78–81. doi:10.1109/MC.2016.89.
- [5] M. Hammoudeh, G. Epiphaniou, S. Belguith, D. Unal, B. Adebisi, T. Baker, A. Kayes, P. Watters, A service-oriented approach for sensing in the internet of things: intelligent transportation systems and privacy use cases, *IEEE Sensors Journal*.
- [6] Y. Yusoff, R. Ismail, Z. Hassan, Common phases of computer forensics investigation models, *International Journal of Computer Science and Information Technology* 3 (3) (2011) 17–31. doi:10.5121/ijcsit.2011.3302.
- [7] N. Kaaniche, Cloud data storage security based on cryptographic mechanisms, Ph.D. thesis, Institut National des Télécommunications (2014).
- [8] A. Bentajer, M. Hedabou, K. Abouelmehdi, S. Elfezazi, Cs-ibe: a data confidentiality system in public cloud storage system, *Procedia Computer Science* 141 (2018) 559–564.
- [9] S. Belguith, N. Kaaniche, M. Hammoudeh, T. Dargahi, Proud: Verifiable privacy-preserving outsourced attribute based signcryption supporting access policy update for cloud assisted iot applications, Vol. 111, 2020, pp. 899–918.
- [10] M. Li, S. Yu, Y. Zheng, K. Ren, W. Lou, Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption, *IEEE transactions on parallel and distributed systems* 24 (1) (2012) 131–143.
- [11] Z. Wan, R. H. Deng, et al., Hasbe: A hierarchical attribute-based solution for flexible and scalable access control in cloud computing, *IEEE transactions on information forensics and security* 7 (2) (2011) 743–754.
- [12] H. A. Al Hamid, S. M. M. Rahman, M. S. Hossain, A. Almogren, A. Alamri, A security model for preserving the privacy of medical big data in a healthcare cloud using a fog computing facility with pairing-based cryptography, *IEEE Access* 5 (2017) 22313–22328.
- [13] R. M. Daniel, E. B. Rajasingh, S. Silas, Analysis of hierarchical identity based encryption schemes and its applicability to computing environments, *Journal of information security and applications* 36 (2017) 20–31.
- [14] Y. Xu, H. Zhong, J. Cui, An improved identity-based multi-proxy multi-signature scheme, *Journal of Information Hiding and Multimedia Signal Processing* 7 (2) (2016) 343–351.
- [15] Y. Sun, W. Zheng, An identity-based ring signcryption scheme in ideal lattice., *J. Netw. Intell.* 3 (3) (2018) 152–161.
- [16] B. Lynn, On the implementation of pairing-based cryptosystems, Ph.D. thesis, Stanford University Stanford, California (2007).
- [17] A. Kumar, A. Jerome, G. Khanna, H. Veladanda, H. Ly, N. Chai, R. Andrews, Elliptic curve cryptography (ecc). certificates performance analysis, https://www.websecurity.digicert.com/content/dam/websitesecurity/digitalassets/desktop/pdfs/whitepaper/Elliptic_Curve_Cryptography_ECC_WP_en_us.pdf, (Accessed on 03/04/2021) (May 2013).
- [18] P. Shynu, R. Nadesh, V. G. Menon, P. Venu, M. Abbasi, M. R. Khosravi, A secure data deduplication system for integrated cloud-edge networks, *Journal of Cloud Computing* 9 (1) (2020) 1–12.
- [19] B. Schneier, K. Seidel, S. Vijayakumar, A worldwide survey of encryption products, Berkman Center Research Publication (2016-2).
- [20] M. Focus, Scalable email security | voltage securemail cloud | micro focus, <https://www.microfocus.com/en-us/products/cloud-email-encryption/overview>, (Accessed on 03/04/2021) (2019).
- [21] X. Boyen, General ad hoc encryption from exponent inversion ibe, in: Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 2007, pp. 394–411.
- [22] D. Boneh, M. Franklin, Identity-based encryption from the weil pairing, *SIAM journal on computing* 32 (3) (2003) 586–615.
- [23] C. Cocks, An identity based encryption scheme based on quadratic residues, in: IMA international conference on cryptography and coding, Springer, 2001, pp. 360–363.
- [24] R. Sakai, M. Kasahara, et al., Id based cryptosystems with pairing on elliptic curve., *IACR Cryptol. ePrint Arch.* 2003 (2003) 54.
- [25] S. D. Galbraith, K. G. Paterson, N. P. Smart, Pairings for cryptographers, *Discrete Applied Mathematics* 156 (16) (2008) 3113–3121.
- [26] A. Kate, I. Goldberg, Distributed private-key generators for identity-based cryptography, in: International Conference on Security and Cryptography for Networks, Springer, 2010, pp. 436–453.
- [27] H. Okano, K. Emura, T. Ishibashi, T. Ohigashi, T. Suzuki, Implementation of a strongly robust identity-based encryption scheme over type-3 pairings, *International Journal of Networking and Computing* 10 (2) (2020) 174–188.
- [28] D. Chu, J. Lin, F. Li, X. Zhang, Q. Wang, G. Liu, Ticket transparency: Accountable single sign-on with privacy-preserving public logs, in: International Conference on Security and Privacy in Communication Systems, Springer, 2019, pp. 511–531.
- [29] M. Kumar, S. Chand, Eski-ibe: Efficient and secure key issuing identity-based encryption with cloud privacy centers, *Multimedia Tools and Applications* 78 (14) (2019) 19753–19786.
- [30] S.-Y. Tan, K.-W. Yeow, S. O. Hwang, Enhancement of a lightweight attribute-based encryption scheme for the internet of things, *IEEE Internet of Things Journal* 6 (4) (2019) 6384–6395.
- [31] X. Zhang, Y. Tang, H. Wang, C. Xu, Y. Miao, H. Cheng, Lattice-based proxy-oriented identity-based encryption with keyword search for cloud storage, *Information Sciences* 494 (2019) 193–207.
- [32] R. Sakai, K. Ohgishi, M. Kasahara, Cryptosystems based on pairing, in: The 2000 symposium on cryptography and information security, IEICE, 2000, pp. 26–82.
- [33] Y. Chen, Q. Huang, Z. Zhang, Sakai-ohgishi-kasahara identity-based non-interactive key exchange revisited and more, *Int. J. Inf. Secur.* 15 (1) (2016) 15–33. doi:10.1007/s10207-015-0274-0.
- [34] D. P. J. Taylor, H. Mwiki, A. Dehghantana, A. Akibini, K. K. R. Choo, M. Hammoudeh, R. Parizi, Forensic investigation of cross platform massively multiplayer online games: Minecraft as a case study, *Science Justice* 59 (3) (2019) 337–348. doi:<https://doi.org/10.1016/j.scijus.2019.01.005>
- [35] H. Arshad, A. B. Jantan, O. I. Abiodun, Digital Forensics: Review of Issues in Scientific Validation of Digital Evidence, *Journal of Information Processing Systems* 14 (2) (2018) 346–376. doi:10.3745/JIPS.03.0095. URL <https://doi.org/10.3745/JIPS.03.0095>
- [36] IEEE, Ieee 1363.3-2013 - ieee standard for identity-based cryptographic techniques using pairings, https://standards.ieee.org/standard/1363_3-2013.html, (Accessed on 03/04/2021) (2013).
- [37] I. F. Blake, G. Seroussi, N. P. Smart, Advances in elliptic curve cryptography, Vol. 317, Cambridge University Press, 2005.
- [38] B. Moller, T. Duong, K. Kotowicz, ssl-poodle.pdf, <https://www.openssl.org/~bodo/ssl-poodle.pdf>, (Accessed on 03/04/2021) (September 2014).
- [39] D. Adrian, K. Bhargavan, Z. Durumeric, P. Gaudry, M. Green, J. A. Halderman, N. Heninger, D. Springall, E. Thomé, L. Valenta, et al., Imperfect forward secrecy: How diffie-hellman fails in practice, in: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, 2015, pp. 5–17.
- [40] N. Aviram, S. Schinzel, J. Somorovsky, N. Heninger, M. Dankel, J. Steube, L. Valenta, D. Adrian, J. A. Halderman, V. Dukhovni, et al., {DROWN}: Breaking {TLS} using sslv2, in: 25th {USENIX} Security Symposium ({USENIX} Security 16), 2016, pp. 689–706.
- [41] L. C. Chen, Z. Smart, Np 2007. identity-based key agreement protocols from pairings, *International Journal of Information Security*.
- [42] C. Langos, M. Giancaspro, Does cloud storage lend itself to cyberbullying?, *IEEE Cloud Computing* 2 (5) (2015) 70–74.
- [43] A. Joux, A one round protocol for tripartite Diffie-Hellman, in: W. Bosma (Ed.), ANTS 2000, Vol. 1838 of LNCS, Springer-Heidelberg, 2000, pp. 385–394.
- [44] M. C. SDK, Multiprecision integer and rational arithmetic cryptographic library, <https://github.com/miracl/MIRACL>, (Accessed on 03/04/2021).

- 1185 [45] S. Belguith, N. Kaaniche, M. Hammoudeh, Analysis of attribute-based cryptographic techniques and their application to protect cloud services, Transactions on Emerging Telecommunications Technologies n/a (n/a) e3667, e3667 ett.3667. [doi:https://doi.org/10.1002/ett.3667](https://doi.org/10.1002/ett.3667)