




Article

Secure Token–Key Implications in an Enterprise Multi-Tenancy Environment Using BGV–EHC Hybrid Homomorphic Encryption

Pooja Dhiman ¹, Santosh Kumar Henge ¹, Rajakumar Ramalingam ², Ankur Dumka ^{3,4}, Rajesh Singh ⁵, Anita Gehlot ⁵, Mamoon Rashid ^{6,*}, Sultan S. Alshamrani ⁷, Ahmed Saeed AlGhamdi ⁸ and Abdullah Alshehri ⁹

 check for updates

Citation: Dhiman, P.; Henge, S.K.; Ramalingam, R.; Dumka, A.; Singh, R.; Gehlot, A.; Rashid, M.; Alshamrani, S.S.; AlGhamdi, A.S.; Alshehri, A. Secure Token–Key Implications in an Enterprise Multi-Tenancy Environment Using BGV–EHC Hybrid Homomorphic Encryption. *Electronics* **2022**, *11*, 1942. <https://doi.org/10.3390/electronics11131942>

Academic Editors: Hamed Taherdoost, Seyed Reza Shahamiri, Kamaljeet Sandhu, Basit Qureshi, Hazra Imran and Salimur Choudhury

Received: 17 May 2022

Accepted: 19 June 2022

Published: 21 June 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

- ¹ School of Computer Applications, Lovely Professional University, Phagwara 144402, India; poojadhiman25@yahoo.co.in (P.D.); hingesanthosh@gmail.com (S.K.H.)
 - ² Department of Computer Science and Technology, Madanapalle Institute of Technology & Science, Madanapalle 517325, India; ramukshare@gmail.com
 - ³ Department of Computer Science and Engineering, Women Institute of Technology, Dehradun 248007, India; ankurumka2@gmail.com
 - ⁴ Department of Computer Science and Engineering, Graphic Era Deemed to be University, Dehradun 248001, India
 - ⁵ Uttaranchal Institute of Technology, Uttaranchal University, Dehradun 248007, India; rajeshsingh@uttaranchaluniversity.ac.in (R.S.); anita.ri@uttaranchaluniversity.ac.in (A.G.)
 - ⁶ Department of Computer Engineering, Faculty of Science and Technology, Vishwakarma University, Pune 411048, India
 - ⁷ Department of Information Technology, College of Computer and Information Technology, Taif University, P.O. Box 11099, Taif 21944, Saudi Arabia; susamash@tu.edu.sa
 - ⁸ Department of Computer Engineering, College of Computer and Information Technology, Taif University, P.O. Box 11099, Taif 21994, Saudi Arabia; asjannah@tu.edu.sa
 - ⁹ Department of Information Technology, Al Baha University, P.O. Box 1988, Al Baha 65731, Saudi Arabia; aashehri@bu.edu.sa
- * Correspondence: mamoon.rashid@vupune.ac.in; Tel.: +91-7814346505

Abstract: Authentication, authorization, and data access control are playing major roles in data security and privacy. The proposed model integrated the multi-factor authentication–authorization process with dependable and non-dependable factors and parameters based on providing security for tenants through a hybrid approach of fully homomorphic encryption methodology: the enhanced homomorphic cryptosystem (EHC) and the Brakersky–Gentry–Vaikuntanathan (BGV) scheme. This research was composed of four major elements: the fully homomorphic encryption blended schemes, EHC and BGV; secure token and key implications based on dependable and non-dependable factors; an algorithm for generating the tokens and the suitable keys, depending on the user’s role; and the execution of experimental test cases by using the EHC algorithm for key and token generation, based on dependable and non-dependable parameters and time periods. The proposed approach was tested with 152 end-users by integrating six multi-tenants, five head tenants, and two enterprise levels; and achieved a 92 percent success rate. The research integrated 32-bit plain text in the proposed hybrid approach by taking into consideration the encryption time, decryption time, and key generation time of data transmission via cloud servers. The proposed blended model was efficient in preventing data from ciphertext attacks and achieved a high success rate for transmitting data between the multi-tenants, based on the user-role-user type of enterprise cloud servers.

Keywords: Session Initiation Time (SIT); Type of Tenant (TTenant); Type of Data (TData); Authentication Type (AT); On-demand OTP (OOTP); Hybrid Key (HK); OS Salting Key (OSSK)

1. Introduction

Multi-tenancy is a software architecture in which multiple customers (tenants) can coordinate with each other in a single cloud environment [1]. Service can be provided in a customized approach to the tenants according to their roles and responsibilities. Multi-tenancy is a combination of virtualization, resource sharing, and authentication [2,3].

Virtualization + Resource Sharing + Authentication = Multi-tenancy

In a multi-tenant environment, it is difficult to provide customized services to various tenants with complete confidentiality and protection of their data. A tenant can be a single user, an organization, or a group of organizations [2]. The communication between the tenants should be designed to secure data in transmission. A homomorphic encryption algorithm is used for encrypting data on the cloud and computations can be done on the encrypted data without decryption [4,5]. Hence, multi-tenancy provides a secure cloud environment by allowing operations on encrypted data. Third parties can work with the data according to the user's requirements; safety and privacy are maintained in this scenario [6,7].

There are three types of homomorphic algorithm: the partial homomorphic encryption algorithm (PHE), the somewhat homomorphic encryption algorithm (SWHE), and the fully homomorphic encryption algorithm (FHE). A PHE supports only one type of computation at a time—either addition or multiplication. RSA(*), Elgamal(*), and Paillier(+) are examples of this category. An SWHE supports a limited number of computations on the different types of operations, such as multiplicative computations and additive computations at the same time but with a limited number of operations, such as learning with errors and ring learning with errors [8]. The FHE scheme supports an unlimited number of any kinds of operations at any instant of time—any number of operations can be performed on ciphertext by mixed operators. FHE can be an additive, multiplicative, or mixed operation [9,10].

In the proposed model, two FHE schemes were used: an enhanced homomorphic cryptosystem (EHC) and the Brakersky–Gentry–Vaikuntanathan scheme (BGV). Both of these schemes worked together to overcome the disadvantages of each, resulting in a more secure cloud environment.

The multi-tenant cloud environment involves one enterprise communicating with another enterprise. The authorization and authentication process in an enterprise requires multi-factor parameters for authentication purposes. For example, apart from an account's username and password, a one-time password (OTP) is sent to an email or mobile device for login, and that password is mapped with a salting technique that shuffles the characters, based on the operating system. In the proposed model, an additional key—an on-demand OTP—was generated, which was active only in the event of an attack.

We found that the homomorphic algorithms, EHC and BGV, formed a better combination than alternatives, as they each offset the disadvantages of the other. The EHC algorithm can work on static data only, while the BGV algorithm can work on both static and dynamic data. BGV requires more storage, while EHC requires less power and storage. EHC is indistinguishable under a chosen ciphertext attack, while BGV is indistinguishable only under chosen-plaintext attack [11,12].

In our research, in a multi-tenant cloud environment, the two FHE schemes were used, together with additional security parameters [13]. These parameters were categorized into dependable and non-dependable parameters. The FHE schemes alone were not sufficient in providing security against upcoming threats and could not provide both access control and data privacy at the same time. Hence, we developed a mechanism to control unauthorized access by generating an additional hidden key in case of an attack, applying parity mapping in the event of an attack. This additional key was sent, together with the public and private keys that were generated using the EHC and BGV encryption algorithms.

Our research integrated the multi-factor authentication–authorization process with dependable and non-dependable factors and parameters, based on providing security to

tenants via the EHC–BGV hybrid approach of the fully homomorphic encryption methodology. Our research was composed of four major elements:

- A description of the fully homomorphic encryption blended schemes, EHC and BGV;
- Secure token and key implications based on dependable and non-dependable factors;
- An algorithm for generating the token and the suitable keys depending on the user's role; and
- The execution of experimental test cases using the EHC algorithm for key and token generation. based on dependable and non-dependable parameters and the time required.

This article is organized as follows: Section 1 includes the research objectives. Section 2 provides the relevant literature together with the background to the research. Section 3 includes the proposed methodology that has been designed based on the BGV–EHC-homomorphic encryption techniques, together with the fully homomorphic encryption blended schemes,, secure token and key implications based on dependable and non-dependable factors, and the algorithm for generating the tokens and the suitable keys, depending on the user's role. Section 4 contains the experimental analysis and results. Section 5 states the conclusions and future directions.

2. Related Works

Previous researchers incorporated various techniques for providing security and privacy to multi-tenants in single-multi enterprises. Some researchers integrated ciphertext policy attribute-based encryption (CP-ABE) [14,15]. Others used proxy re-encryption, key policy attribute-based encryption (KP-ABE) [16], a JSON-based model with HTTPS/TLS transmission [17], an SLA-oriented multi-tenant hybrid scheme [18], a fully homomorphic encryption scheme such as the enhanced algebra homomorphic encryption (AHEE) scheme [19] or a, non-interactive exponential homomorphic encryption algorithm [20]. Some researchers used validation- and verification-based models for data access and control among the various supportive tenants [15,21].

Wang et al. [18] suggested using a performance isolation technique among tenants, as sharing resources among tenants results in service performance competition. The methodology used in that paper included an SLA-oriented multi-tenant hybrid scheme. Container technology was used for performance isolation. The experiments resulted in a better and more cost-effective solution for performance isolation competition. The future scope of that study included additional factors and parameters based on security among tenants.

Sammata and Parthiban [22] analyzed a fully homomorphic encryption scheme, an enhanced algebra homomorphic encryption scheme (AHEE), based on updated ElGamal encryption. The scheme was used to process medical data in hospitals securely and privately. By checking patient history, a possible future health issue could be identified so the patient could take appropriate medications in advance. The cloud administrators did not have any knowledge of the patients' medical information. The AHEE took less time for encryption and decryption. The results of that paper could be enhanced by adding biometrics such as a patient's fingerprint, a pupil tracker, etc.

Sun [16] analyzed various technologies, such as access control, cipher-text policy attribute-based encryption (CP-ABE), proxy re-encryption, key policy attribute-based encryption (KP-ABE), multi-tenancy, and other related technologies in terms of cloud security. That paper suggested the inclusion of updated policies and the integration of new technologies to protect data. The practical implementation of the suggestions was left to future researchers.

Kanagavalli and Vagdevi [19] compared the BGV homomorphic scheme with the RSA and AHEE homomorphic schemes. They found that the encryption and decryption times required by BGV were comparatively less than those required by RSA and AHEE. Further research could be done to explore ways of storing data in less space, as BGV requires a huge amount of memory and sometimes becomes difficult to manage in real-time applications.

Ethelbert et al. [17] proposed a JSON-based model for an improved authentication process. The authors used a dual authentication process and suggested using HTTPS/TLS transmission for authorization purposes. However, the dual authentication process was not sufficient for upcoming threats; researchers should include additional modes for authentication.

Awadallah et al. [23] proposed a verification scheme to validate homomorphic computations over a finite integer field. A homomorphic encryption algorithm was not enough to provide data integrity, as computations could be swapped with other computations. Hence, the authors incorporated a verifiable scheme for validating the homomorphic computations. Although their proposed scheme added overhead in managing the increased storage requirements and computations, the authors regarded this overhead as acceptable.

Bhatia and Dave [24] concluded that implementing FHE schemes is difficult and that applications using FHE lack necessary security. The authors proposed “privacy homomorphism” that allows multi-party computation securely. A comparison of various algorithms was presented. It was found that elliptic curve cryptography (ECC) provides better results and can be used for sensitive applications with FHE schemes.

Oladunni and Sharma [25] suggested using a homomorphic encryption algorithm for securing the cloud. A homomorphic algorithm was categorized into three types, based on the number of operations. A partial homomorphic algorithm performed either addition or multiplication at one time; a somewhat homomorphic algorithm allowed both addition and multiplication at the same instant of time, but with limited operation, while a fully homomorphic algorithm allowed an unlimited number of operations on the ciphertext. RSA, Paillier, and ElGamal are examples of partial homomorphic algorithms; R-LWE and LWE are examples of somewhat homomorphic algorithms, as they are based on noise; and EHC, BGV, AHEE, and NEHE are examples of fully homomorphic algorithms. The cloud can be protected using a fully homomorphic encryption algorithm, as it allows, computation on ciphertext which means that there is no need to decrypt data.

Al-Mashhadi and Khalf [26] proposed a hybrid homomorphic cryptosystem for transferring colored images securely on the public cloud. ElGamal encryption and EHC algorithms securely transferred the colored images based on the block pixel position. The model was considered secure and provided improved security on the public cloud.

Parmar et al. [27] surveyed the fully homomorphic schemes, EHC, BGV, NEHE, and AHEE. The advantages and disadvantages of the number of keys used in each scheme and the implementation details were presented. Saeed [28] proposed a three-layer cryptographic algorithm using the AES, ECC, and RSA methods. The author implemented the RSA–ECC algorithm in Java NetBeans and compared it with the RSA–AES combination; is the author found that the encryption and decryption times in the RSA–ECC method were comparatively less than those of other methods.

Most of the previous approaches used a token-based model and a dual authentication method [29–32]. Some authors used fully homomorphic schemes and introduced the tenant isolation concept [21,33]. However, existing methodologies and parameters are not sufficient in securing a multi-tenant cloud environment.

Therefore, we developed a new multi-factor authentication approach with a customized tenant environment, within which authentication is based on the role of the user. A combination of the fully homomorphic encryption algorithms EHC and BGV were used for the generation of tokens and keys.

3. Methodology

In the architecture proposed by this study (refer Figure 1), one main server is used in each organization or enterprise. A total of two main servers and two communicating servers are required for sending packets or sharing data. There are four departments in each enterprise, for a total of eight departments, with 16 users in the first enterprise and 15 in the second one. Two firewall routers, either a Cisco router or a simple checkpoint device, are attached between the main server and the web server. Their main responsibility is to

protect data from unauthorized access. Figure 1 shows the enterprise-level architecture in a multi-tenant environment.

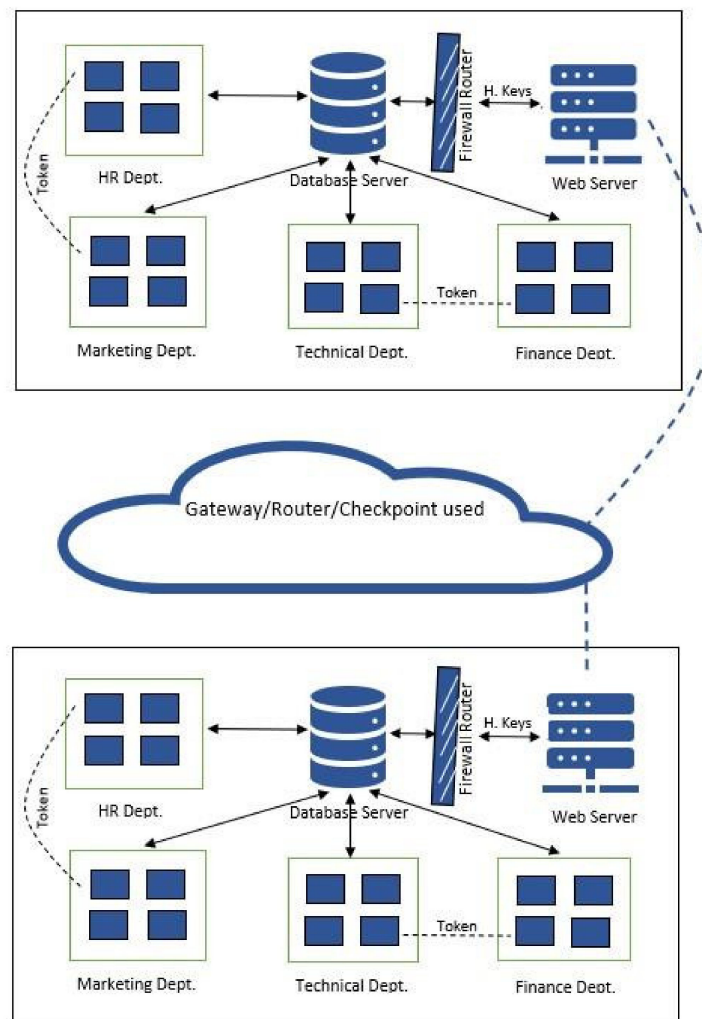


Figure 1. Enterprise-level multi-tenancy environment for data access control, authorization, and authentication.

The main server or database server is responsible for internal communication among tenants. Tokens are used for internal verification, as in a single enterprise multiple tenants are working on the same LAN. The web server is used to communicate to different enterprises or outside the organization, because sending data on the cloud using tokens is not sufficient in terms of security. A key is generated using a homomorphic encryption algorithm which is hidden by default, to be activated only in case of attacks on the packets. A firewall router, such as a Cisco or checkpoint device, ensures data privacy and protects the data from attacks. As the web server is the communicating server, the IP address is known, and hence an extra layer of security is required; in contrast, the main server works in the background in a more secure environment, where the IP addresses are not known to intruders.

3.1. The Fully Homomorphic Encryption Blended Schemes EHC and BGV-Based Environment

A fully homomorphic encryption scheme ensures that data computation is done in an encrypted state without the need to decrypt the data [34]. Hence, it provides better security for cloud data and services. Keys are generated using a combination of the fully homomorphic encryption schemes, EHC and BGV.

This study's methodology used EHC and BGV algorithms for key generation. The enhanced homomorphic cryptosystem (EHC) requires less time for the decryption process and consumes low energy. The storage requirement is comparatively less than those the other schemes. EHC is IND-CCA secure, which means that an attacker cannot distinguish the keys for a chosen ciphertext.

Four keys were generated: one public key and three private keys. EHC is static and supports only static data. The Brakerski-Gentry-Vaikuntanathan (BGV) scheme reduces the noise so that the ciphertext can be managed without a bootstrapping technique [35,36]. Modulus switching is used in BGV, supporting both types of data, static and dynamic. A small overhead of storage is present in this algorithm, which is indistinguishable under a chosen-plaintext attack (IND-CPA secure) and does not provide security for CCA. Two keys are generated in BGV. EHC and BGV are used in combination to overcome the disadvantages of each algorithm, resulting in a better algorithm with additional security.

In this study's proposed model, EHC is used to generate the token key, which are used by an insider user for internal communication within the organization. As internal communication is considered safe, and no other keys are generated. The token expires after a specific time, depending on the role of the user in the organization.

A token may have different access controls for a file. For example, an accounts person may have different rights to a file than an IT person. Hence, token lifetime provides greater security within an organization. Another example is a banking institution, wherein the login session expires after some time of user inactivity. In the same way, in the current scenario, the token expires after a specified time.

For outsider users, the BGV algorithm is used, as it is dynamic. In the case of a trusted user, the BGV algorithm generates a hybrid key based on the generation rules for the keys. A hybrid key is a combination of private and public keys. Apart from this hybrid key, an additional 4-bit or 8-bit key is generated, which is hidden by default or is in a disabled state. It becomes active only in case of an attack.

For example, suppose two private keys and one public key are generated using FHE. A new 4-bit hidden key will be generated using the parity bit method on the keys that have already been generated. This hidden key is sent to the user, together with the other public and private keys. The hidden key becomes active if the packets are attacked.

Now, if the user is a partial user, then the user may not be trusted; hence, the token expiration concept is introduced in this situation. The session for a partial user remains active for some specific number of days. In addition, for an anonymous or guest user, the session remains active only for a specific number of hours, and the security for a guest user is as least as good as that of the other categories. Hence, BGV and EHC combine to provide a secured cloud environment.

3.2. Secure Token and Key Implications Based on Dependable and Non-Dependable Factors

In our proposed system, the tokens and keys are generated on the basis of the dependable and non-dependable parameters. Some of the dependable parameters are OTP, password, KLH, and KLD, while non-dependable parameters include open one-time password (OOTP), salting, and so on (refer Figure 2). The communication process depends on the role of the user; a user can be an insider or an outsider. An insider belongs to the organization and is provided with all of the access permissions, while an outsider does not belong to the organization and is considered unsafe. Hence, an extra level of security should be added to protect the cloud environment. For outsider users, security can be added. If a user is trusted, access can be provided to the user, and apart from the two-factor authentication (i.e., a combination of username, password, and OTP), salting is added [37]. An additional key is generated that will be active in case of an attempt by an attacker to access the data. For an untrusted user, the token key should expire after a period of time. A partial user is provided access for a certain number of days, while a temporary user is given access only for a few hours, as such use is considered to be the least secure mode of transmission.

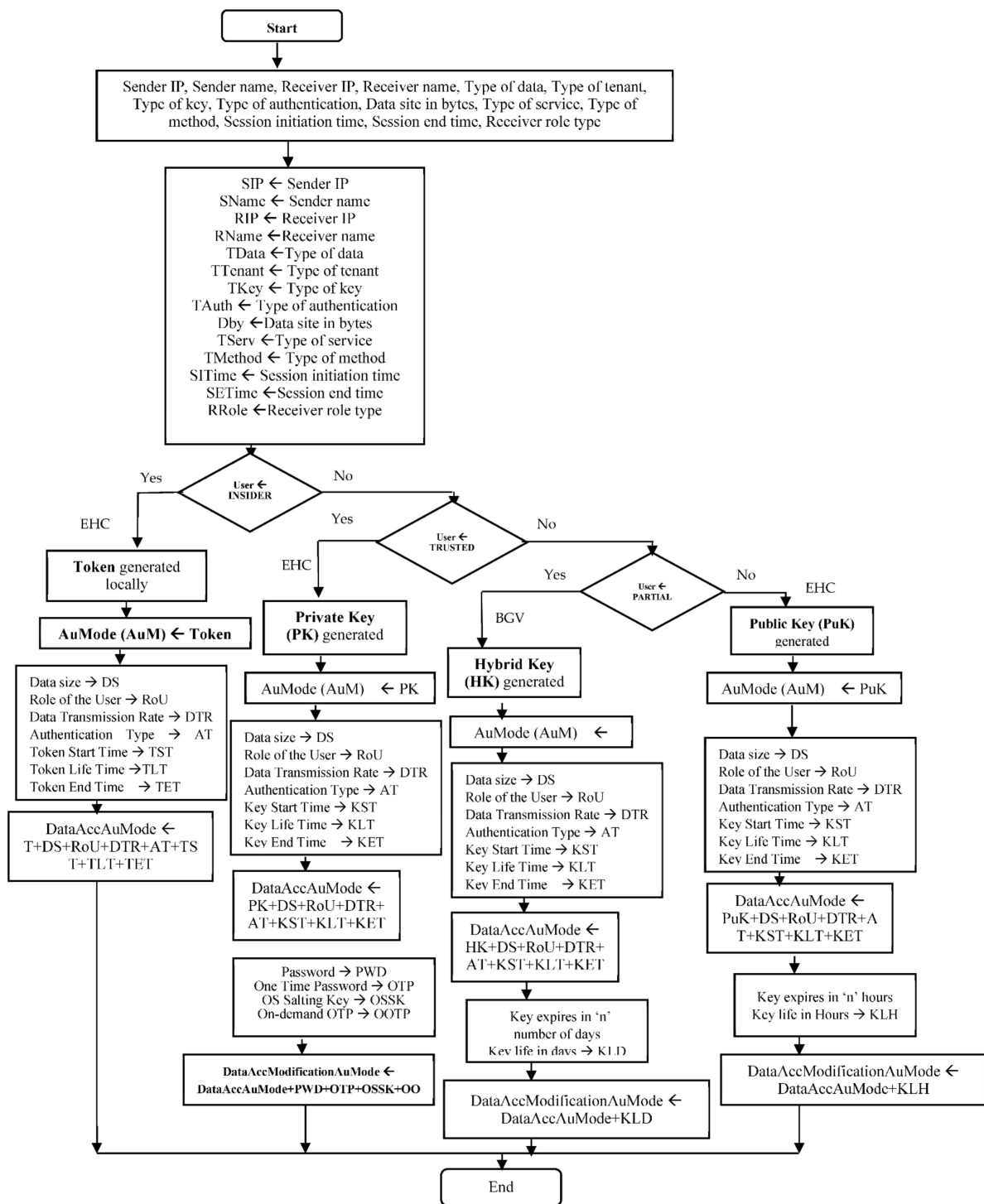


Figure 2. Steps for generating the tokens and the associated keys depending on the role of the user.

The scenario shown in Figure 2 is explained in detail as follows:

First, the required parameters are sent to initiate the transmission channel. These parameters are sender IP, sender name, receiver IP, receiver name, type of data, type of tenant, type of key, type of authentication, data site in bytes, type of service, type of method, session initiation time, session end time, and type of receiver role. The basic information about the sender, receiver, authentication type, keys type, and service type is shared. Then, the user’s type or role is verified; if the user is an insider, then only a token is generated, because the communication is within the organization and is considered safe.

The authentication mode requires a token, and in the next step, the parameters sent for transaction processing are sender IP, sender name, receiver IP, receiver name, type of data, type of tenant, type of key, type of authentication, data site in bytes, type of service, type of method, session initiation time, session end time, type of receiver role. The next step requires evaluating the data access authentication mode, which is a combination of the parameters shared for transaction-processing in the previous step and the token as randomly generated. No modification of the parameters is required because an insider user has all the access permissions, and no changes are needed. If the user is an outsider, this means that the user does not belong to the organization; hence, an extra layer of protection or security is required. In the outsider category, there are subcategories for a trusted user, a partially trusted user, and a guest user.

A trusted user is an authorized user that requires a higher level of security than an outsider. The authenticity of the trusted user is verified using different methods. A private key is generated and the authentication mode uses a private key only; no token is required. To access the data, parameters are required for the authentication process, such as data site, data transmission rate, authentication type, key start time, key lifetime, and key end time. For the login process, an OTP is generated together with the username and password and sent to the user's mobile number. Salting is provided by the operating system; it is a process of adding some characters with the original password and shuffling the characters so that it becomes difficult for an attacker to determine the password. In addition to these three login security methods, an additional key OOTP (i.e., a one-demand OTP) is generated only if an attacker tries to attack the packets. It will be active only in the attacking scenario; otherwise, it remains disabled. The OOTP is a 4-bit or 8-bit key generated by the parity bit method. It provides an extra layer of security for the packets in transmission.

Another user type in the outsider untrusted category is the partial user, for which the token expiration concept is introduced. The session will be active for a certain number of days and the token will expire after the specified time. First, the role of the user is checked. If the user is partial, a hybrid key is generated. A hybrid key is a combination of the private and public keys generated on the basis of certain rules and conditions.

The authentication mode uses a hybrid key, and the data access authentication mode requires a hybrid key and parameters such as data site, data transmission rate, authentication type, key start time, key lifetime, and key end time. For a partial user, the token should expire automatically after "n" number of days, due to security concerns. In the data access modification authentication mode, changes take place: the parameters are updated in the database after they are processed. The data access modification authentication mode is a combination of the data access authentication mode parameters and the hybrid key and key life in days. The last case is a type of outsider un-trusted user, which is a guest user. This category of users is considered as the least secure category among all those that are available.

The session will be active for an "n" number of hours, and the token for the current session will thereafter expire. In this category, only a public key is generated, which is the least secure option for accessing data. Attackers can easily access data in this category. A guest user or a temporary user has limited access permissions for a limited time. The parameters passed in this mode are data site, data transmission rate, authentication type, key start time, key lifetime, and key end time. The data access authentication is done with the public key and the parameters that are passed in the previous step. The session will expire after "n" number of hours. Finally, the data access modification authentication mode updates the database with the updated parameter values. The parameters used in this step are a public key and key life in hours. All the other parameters of the data access authentication mode need to be updated.

3.3. Algorithm for Generating the Token and the Suitable Keys Depending on the User's Role

Algorithm for Generating the Token and the Suitable Keys Depending on the User's Role (Algorithm 1).

Algorithm 1. Algorithm for generating the token and the suitable keys depending on the user's role

Step 0: Prepare to send the data.
Initiation taken for sending the data based on parameters such as sender IP, sender name, receiver IP, receiver name, type of data, type of tenant, type of key, type of authentication, data site in bytes, type of service, type of method, session initiation time, session end time, and type of receiver role.

Step 1: Tuning the dependable and non-dependable parameters into variables, such as:
 $SIP \leftarrow$ Sender IP ($SIP \leftarrow sip\{s_1, s_2, s_3, \dots \dots s_n\}$)
 $SName \leftarrow$ Sender name
 $RIP \leftarrow$ Receiver IP ($RIP \leftarrow rip\{r_1, r_2, r_3, \dots \dots r_n\}$)
 $RName \leftarrow$ Receiver name
 $TData \leftarrow$ Type of data ($TData \leftarrow tdata\{td_1, td_2, td_3, \dots \dots td_n\}$)
 $TTenant \leftarrow$ Type of tenant

Step 2: $TKey \leftarrow$ Type of key ($TKey \leftarrow tkey\{tk_1, tk_2, tk_3, \dots \dots tk_n\}$)
 $TAuth \leftarrow$ Type of authentication ($Tauth \leftarrow tauth\{\tau_1, \tau_2, \tau_3, \dots \dots \tau_n\}$)
 $Dby \leftarrow$ Data site in bytes
 $TServ \leftarrow$ Type of service ($TServ \leftarrow tserv\{ts_1, ts_2, ts_3, \dots \dots ts_n\}$)
 $TMethod \leftarrow$ Type of method ($TMethod \leftarrow tmethod\{tm_1, tm_2, tm_3, \dots \dots tm_n\}$)
 $SITime \leftarrow$ Session initiation time ($SITime \leftarrow sitime\{sit_1, sit_2, sit_3, \dots \dots sit_n\}$)
 $SETime \leftarrow$ Session end time ($SETime \leftarrow setime\{set_1, set_2, set_3, \dots \dots set_n\}$)
 $RRole \leftarrow$ Receiver role type ($RRole \leftarrow rrole\{rr_1, rr_2, rr_3, \dots \dots rr_n\}$)

Step 3: User integration: Classification of the user, such that:
 If (user == INSIDER) ($INSIDER \leftarrow insider\{i_1, i_2, i_3, \dots \dots i_n\}$)
 Token (T) generated ($T \leftarrow t\{t_1, t_2, t_3, \dots \dots t_n\}$)
 Tuning the parameters to variables:
 Data size \rightarrow DS ($DS \leftarrow ds\{ds_1, ds_2, \dots \dots ds_n\}$)
 Role of the User \rightarrow RoU ($RoU \leftarrow rou\{rou_1, rou_2, \dots \dots rou_n\}$)

Step 3.1: Data Transmission Rate \rightarrow DTR
 Authentication Type \rightarrow AT ($AT \leftarrow at\{at_1, at_2, \dots \dots at_n\}$)
 Token Start Time \rightarrow TST ($TST \leftarrow tst\{tst_1, tst_2, \dots \dots tst_n\}$)
 Token Lifetime \rightarrow TLT ($TLT \leftarrow tlt\{tlt_1, tlt_2, \dots \dots tlt_n\}$)
 Token End Time \rightarrow TET ($TET \leftarrow tet\{tet_1, tet_2, \dots \dots tet_n\}$)
 $DataAccAuMode \leftarrow T+DS+RoU+DTR+AT+TST+TLT+TET$
 If (user == TRUSTED) ($TRUSTED \leftarrow trusted\{tr_1, tr_2, tr_3, \dots \dots tr_n\}$)
 Private key (PK) generated ($PK \leftarrow pk\{pk_1, pk_2, pk_3, \dots \dots pk_n\}$)
 Tuning the parameters to variables:
 Data size \rightarrow DS ($DS \leftarrow ds\{ds_1, ds_2, \dots \dots ds_n\}$)
 Role of the User \rightarrow RoU ($RoU \leftarrow rou\{rou_1, rou_2, \dots \dots rou_n\}$)
 Data Transmission Rate \rightarrow DTR
 Authentication Type \rightarrow AT ($AT \leftarrow at\{at_1, at_2, \dots \dots at_n\}$)
 Key Start Time \rightarrow KST ($KST \leftarrow kst\{kst_1, kst_2, \dots \dots kst_n\}$)
 Key Lifetime \rightarrow KLT ($KLT \leftarrow klt\{klt_1, klt_2, \dots \dots klt_n\}$)

Step 3.2: Key End Time \rightarrow KET ($KET \leftarrow ket\{ket_1, ket_2, \dots \dots ket_n\}$)
 Accessing data:
 $DataAccAuMode \leftarrow PK+DS+RoU+DTR+AT+KST+KLT+KET$
 Login keys:
 Password \rightarrow PWD
 One Time Password \rightarrow OTP
 OS Salting Key \rightarrow OSSK
 On-demand OTP \rightarrow OOTP {active only in attacking scenario}
 Parameters modified:
 $DataAccModificationAuMode \leftarrow DataAccAuMode +PWD+OTP+OSSK+OOTP$

Algorithm 1. Algorithm for generating the token and the suitable keys depending on the user's role

```

If (user = PARTIAL)
(PARTIAL ← partial {par1, par2, par3, .. parn})
Hybrid key (HK) generated (HK ← hk{hk1, hk2, hk3, ... .. hkn})
Tuning the parameters to variables:
Data size → DS (DS ← ds{ds1, ds2, ... .. dsn})
Role of the User → RoU (RoU ← rou{rou1, rou2, ... .. roun})Data
Transmission Rate → DT
Authentication Type → AT (AT ← at{at1, at2, ... .. atn})
Step 3.3: Key Start Time → KST (KST ← kst{kst1, kst2, ... .. kstn})
Key Lifetime → KLT (KLT ← klt{klt1, klt2, ... .. kltn})
Key End Time → KET (KET ← ket{ket1, ket2, ... .. ketn})
Accessing data:
DataAccAuMode ← HK+DS+RoU+DTR+AT+KST+KLT+KET
Key expiration time:
if (KLT >= KLD) {for KLD as key life time in days}
Key expires
Parameters modified:
DataAccModificationAuMode ← DataAccAuMode + KLD
If (user = ANONYMOUS)
(ANONYMOUS ← an {an1, an2, an3, .. ann})
Public key (PuK) generated
Tuning the parameters to variables:
Data size → DS (DS ← ds{ds1, ds2, ... .. dsn})
Role of the User → RoU (RoU ← rou{rou1, rou2, ... .. roun})
Data Transmission Rate → DTR
Authentication Type → AT (AT ← at{at1, at2, ... .. atn})
Step 3.4: Key Start Time → KST (KST ← kst{kst1, kst2, ... .. kstn})
Key Lifetime → KLT (KLT ← klt{klt1, klt2, ... .. kltn})
Key End Time → KET (KET ← ket{ket1, ket2, ... .. ketn})
Accessing data:
DataAccAuMode ← PuK+DS+RoU+DTR+AT+KST+KLT+KET
Key expiration time:
if (KLT >= KLH) {for KLH as key life in hours}
Key expires
Parameters modified:
DataAccModificationAuMode ← DataAccAuMode + KLH
Step 4: End if
Step 5: End

```

4. Experimental Analysis and Results

A cloud API is designed with the XAMPP tool, which is a cross-platform for developing web applications online. Two servers are integrated with the designed multi-tenant cloud API. One server is the main server or database server; the other is the web server. A database server is used for internal communication in an enterprise or organization, where only tokens are generated to protect a tenant's data. The main server is as safe because it is designed for insider tenants and the IP address of this server is unknown or invisible. The external server used for communication outside the organization is the web server.

In the proposed model, the tomcat server is used as a web server and MySQL is used as a database server. The model is designed for two enterprises with 152 users. The homomorphic algorithms used for key generation are EHC and BGV. EHC generates the public keys, private keys, and tokens, and the BGV algorithm generates the hybrid key.

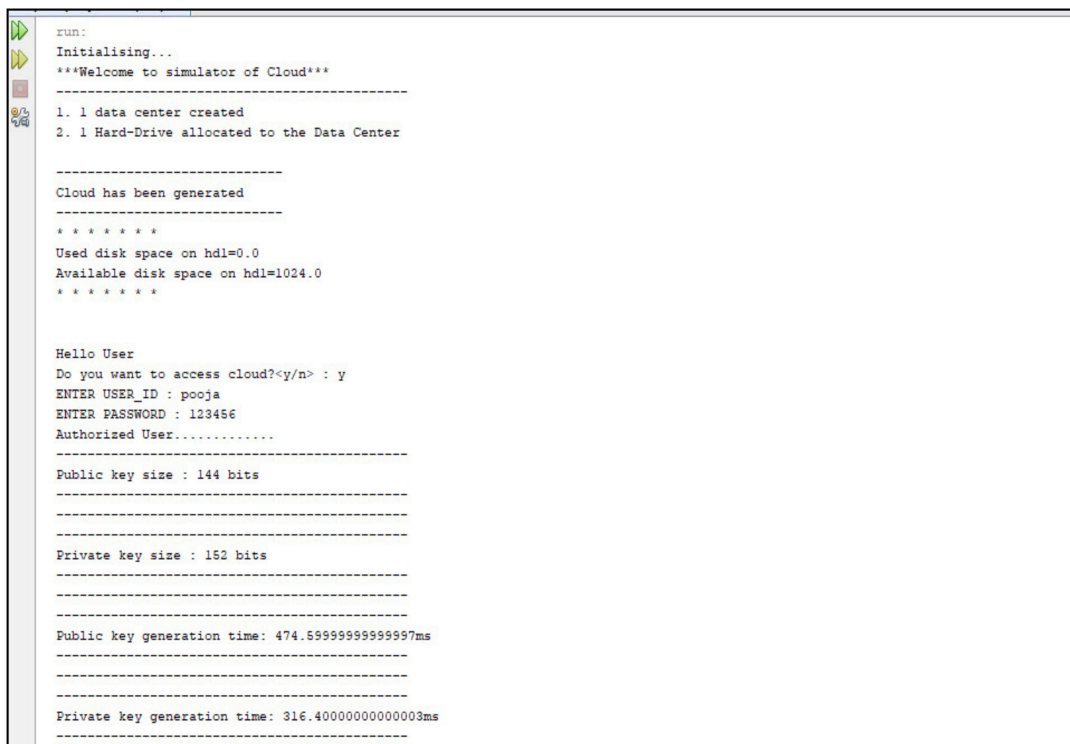
A token is used for internal communication. It expires after a certain time; then, the user or tenant asks for permission from the owner, and the tenant can only access the file again after approval is received. The keys are generated for outsider tenants or for communication outside the organization's premises. The key expiration concept is

introduced at this stage, based on the role of the user. In addition, the hybrid key is generated and remains inactive; it is activated only in an attacking scenario.

Our proposed model provides better security than the current security models, because in addition to username and password, OTP, and salting, it uses a multi-factor authentication method, i.e., a new hidden key that is generated by OOTP (on-demand OTP). OOTP activates at the time of an attack to protect data from unauthorized access. Four keys are used in EHC and two keys are generated in BGV. In addition, one other key is generated. Hence, an additional layer of security is successfully implemented in our proposed model.

4.1. Creating a Multitenant Environment on CloudSim and Generating the Keys Using the EHC Algorithm

In our model, a multitenant cloud environment is created using the CloudSim simulator and Netbeans. The authentication and authorization of tenants are evaluated in accessing a file in the cloud. Our model asks for the credentials of the user or tenant, and only after validating the credentials is a tenant able to access the cloud. The total number of bits required by public and private keys, and the times taken by these keys, are evaluated. Figure 3 shows the time and size of the generation of EHC keys. The time is calculated in nano seconds (ns). The public key size is 144 bits, while the private key size is 152 bits.



```

run:
Initialising...
***Welcome to simulator of Cloud***
-----
1. 1 data center created
2. 1 Hard-Drive allocated to the Data Center
-----
Cloud has been generated
-----
*****
Used disk space on hdl=0.0
Available disk space on hdl=1024.0
*****

Hello User
Do you want to access cloud?<y/n> : y
ENTER USER_ID : pooja
ENTER PASSWORD : 123456
Authorized User.....
-----
Public key size : 144 bits
-----
-----
Private key size : 152 bits
-----
-----
Public key generation time: 474.5999999999997ms
-----
-----
Private key generation time: 316.40000000000003ms
-----

```

Figure 3. Test case with EHC keys' generation time and size; snip for EHC keys generation time and size.

4.2. Token Generation

In our model, a token is generated for internal communication and accessing the file. For better security, the token is sent directly to the registered user or tenant on their email id. The file transfer protocol used for the transmission of the token is FTP. Figure 4 shows a token received on a tenant's email id. This token will be active for only 2 h; after that, a new token is required for access to the same file. A user or tenant cannot access the file with an expired token.

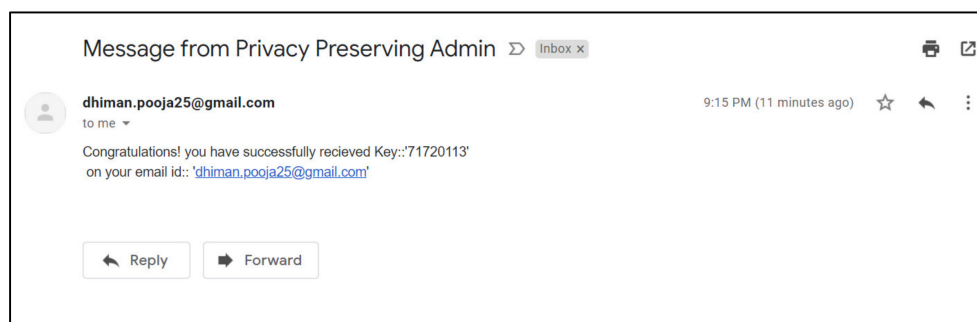


Figure 4. Test case with token generation via email id.

The token expires after a pre-specified time of two hours. After this, the session of the tenant expires, and the tenant is asked to make another request to the owner for access to that file. The tenant makes a further request to the owner for file access, and only after approval and validation from the owner can the file be accessed, using the token sent to the email id of the tenant.

4.3. Discussion

The proposed model uses a combination of fully homomorphic encryption algorithms EHC and BGV for the generation of tokens and keys. A token generated for the security of internal users expires after a certain time. For outside users, keys are generated based on the user’s category. The EHC algorithm is used for token generation and BGV is used to generate a hybrid key. In an outsider environment, the highest security is provided for trusted outsider tenants. In addition to password, salting, and OTP, OOTP (on-demand OTP) is generated. OOTP is a hidden key that becomes active only in the event of attack. For other tenant categories, i.e., for an untrusted user or tenant, two user categories apply: partial and guest/anonymous. For partial users, the hybrid key is generated on the basis of certain production rules, and the session expires after a certain number of days. Guest users are provided the least security; their sessions will expire after a certain number of hours. Thus, security is applied to different layers of tenants, and customized access is provided to tenants according to their roles.

Two keys used by BGV, one private and one public, while one public key and three private keys are generated by EHC. In addition to these keys, one more key, OOTP, is generated when an attack is suspected. This adds an extra layer of protection to cloud data. Hence, the designed multi-factor authentication technique is based on the role of the user, providing better privacy and security for cloud data.

Table 1 shows a comparative analysis of hybrid models proposed in the past. Our proposed model achieved a success rate of 92%, compared with existing models.

Table 1. Comparative analysis of the proposed hybrid model with existing models.

Authors	Application Area	Mode of Security	Homomorphic Encryption Technique	Implicated Environment, Internal Enterprise	Implicated Environment, External Enterprise	Number of Tokens and Keys	Success Rate
Gorti VNKV Subba Rao et al. [17]	Mobile ad hoc networks	Message transmission	EHC	Local tenant	Global tenant	4 keys (1 public, 3 private keys)	100% only if there are more active nodes in each group of networks.
S.V. Suriya Prasad et al. [11]	Cloud security	Data transmission	BGV, EBGV (enhanced BGV)	Local tenant	Global tenant	2 keys (1 public, 1 private key)	Partially executed
Liang Chen et al. [20]	Cloud security	Data transmission	NEHE	Local tenant	Local tenant	4 keys (1 public, 3 private keys)	Partially executed
Pooja Dhiman and Santosh Kumar Henge	Network and cloud security	Data storage and transmission	EHC–BGV	Local loba Local tenant	External enterprise	3 tokens, PuK, HK, PK (6 keys)	92%

The proposed EHC–BGV encryption hybrid model is more efficient than other models in terms of security and privacy, taking into account the data transmission rate, the number of generations of secure tokens and keys, application areas, and implicated environment–internal enterprises such as a local–global–local tenant enterprise and an implicated environment–external enterprise.

Table 2 shows a comparison study of the existing approaches and the proposed hybrid approach, taking into consideration the ciphertext that is used for data transmission via cloud enterprise-level servers. Figure 5 shows a graphical representation of the interpretation, based on the size of the ciphertext compared with those of the existing models, RSA, AHEE, and HE, that adopted a client–cloud provider system.

Table 2. Comparison of the size of the ciphertext in existing approaches and that of the proposed hybrid approach (time in ns).

Plain Text Size in Bits	RSA [38]	AHEE [38]	HE Adoption Client-Cloud Provider System [38]	Proposed EHC–BGV Hybrid Model
8	26	20	18	14
16	28	24	22	22
24	30	28	26	24
32	NA	NA	NA	28

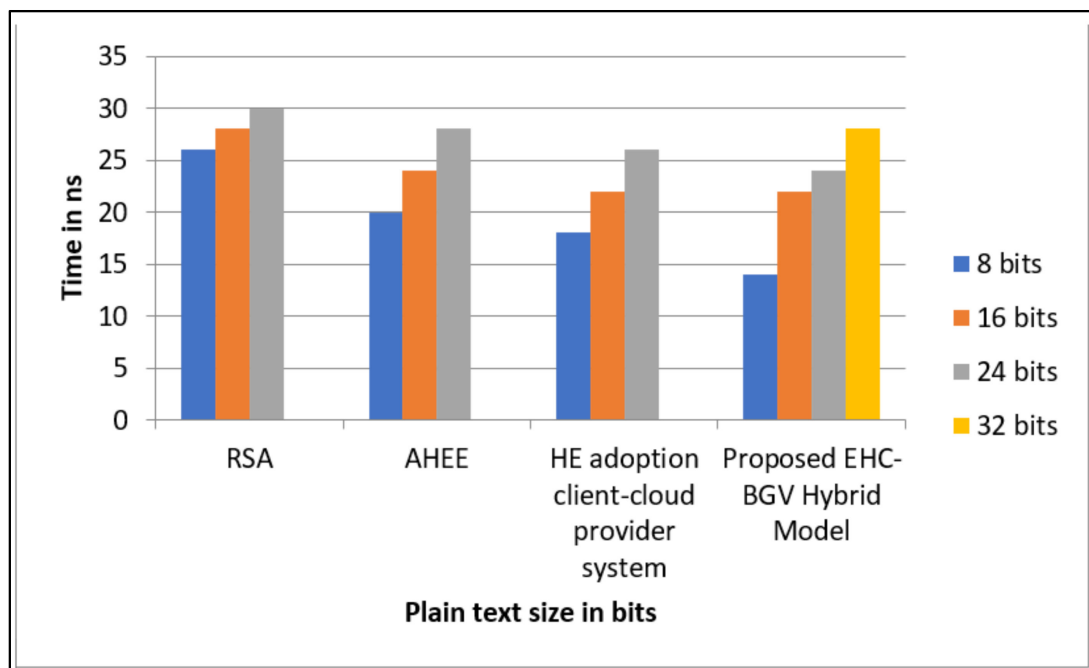


Figure 5. A comparison of the size of the ciphertext in existing approaches with that of the proposed hybrid approach.

A comparison of the existing approaches with the proposed hybrid approach takes into consideration the encryption time of data transmission via the cloud servers, as shown in Table 3. The graphical representation of the table is shown in Figure 6, where the interpretation was done based on the implication of encryption-time compared with those of the existing models, RSA, AHEE and HE, that adopted a client–cloud provider system.

Table 3. Comparison of the time for the implication of encryption in existing approaches and that of the proposed hybrid approach (time in ns).

Plain Text Size in Bits	RSA [38]	AHEE [38]	HE Adoption Client-Cloud Provider System [38]	Proposed EHC-BGV Hybrid Model
8	3.43	1.62	1	0.54
16	8.57	4.07	3.47	2.8
24	12.05	5.67	4.13	3.9
32	NA	NA	NA	5.6

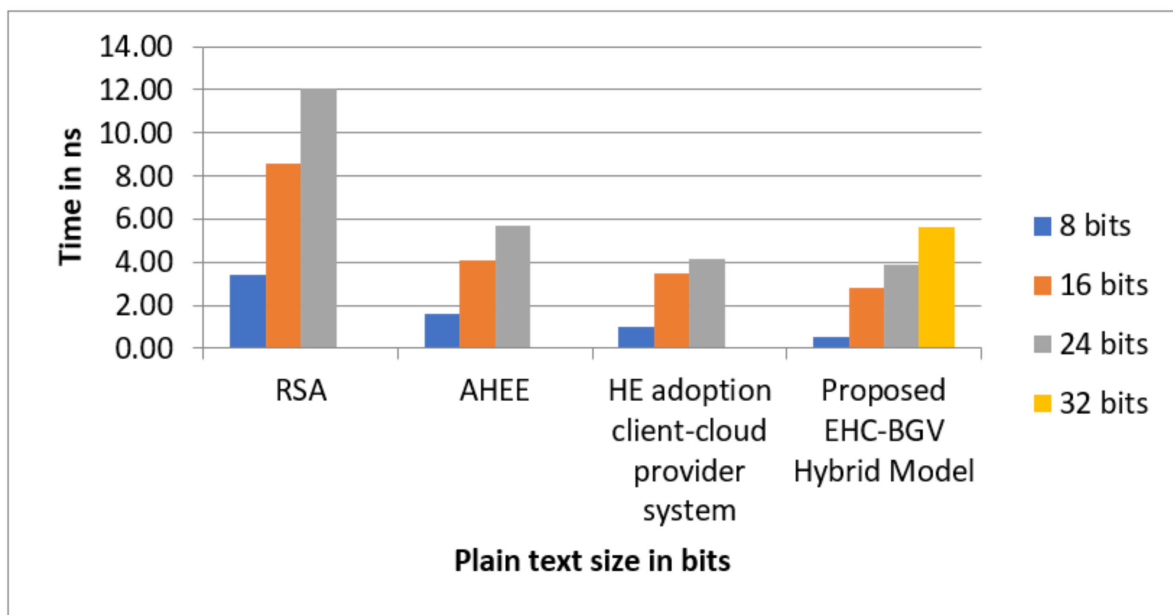


Figure 6. A comparison of the time for the implication of encryption in existing approaches with that of the proposed hybrid approach.

Table 4 shows a comparison study of the existing approaches with the proposed hybrid approach, taking into consideration the decryption times of data transmission via the cloud servers. Figure 7 shows a graphical representation of the interpretation based on the decryption time compared with that of the existing models, RSA, AHEE and HE, which adopted a client–cloud provider system.

Table 4. Comparison of the time for the implication of decryption in existing approaches with that of the proposed hybrid approach (time in ns).

Plain Text Size in Bits	RSA [38]	AHEE [38]	HE Adoption Client-Cloud Provider System [38]	Proposed EHC-BGV Hybrid Model
8	2.56	0.92	0.92	0.87
16	4.42	2.03	1.25	1.10
24	5.96	3.15	2.35	2.12
32	NA	NA	NA	3.27

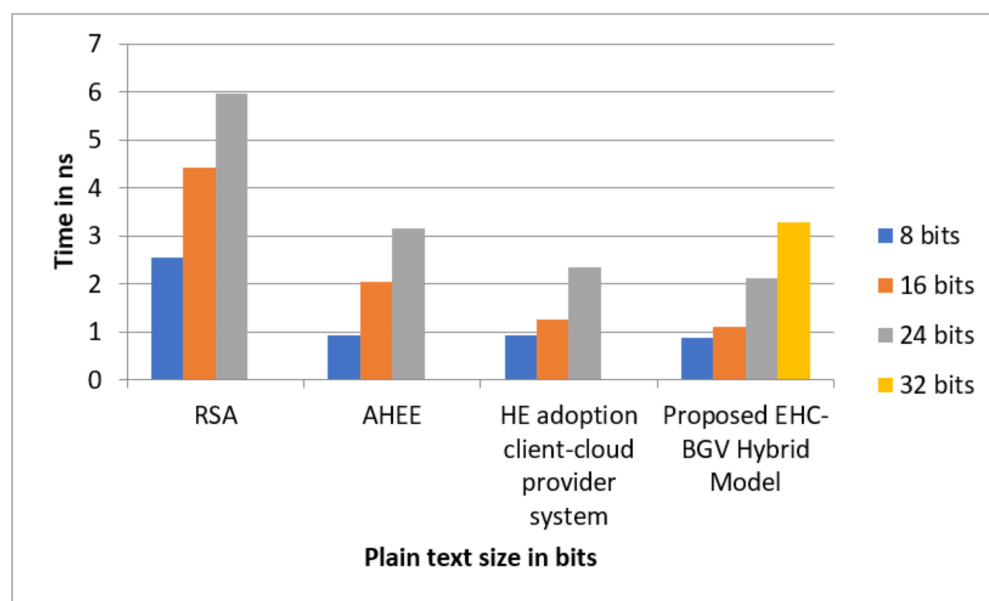


Figure 7. Comparison of the time for the implication of decryption in existing approaches with that of the proposed hybrid approach.

5. Conclusions

This research integrated the multi-factor authentication–authorization process with dependable and non-dependable factors and parameters, based on providing security to tenants via the EHC–BGV hybrid approach of the fully homomorphic encryption methodology. The research was composed of four major elements: a description of the fully homomorphic encryption blended schemes, EHC and BGV; secure token and key implications, based on dependable and non-dependable factors; an algorithm for generating the tokens and the suitable keys, depending on the user’s role; and the execution of experimental test cases with the EHC algorithm for key–token generation, based on dependable and non-dependable parameters and time periods. The proposed blended model was efficient in preventing data from cipher-text attacks and achieving a high success rate for transmitting data between multi-tenants, based on the user-role-user type of enterprise cloud servers. The proposed blended model was efficient in preventing data from cipher-text attacks. We plan on including addition test cases in the future, are based on securing data that is transmitted between multi-tenants of single-multi-single enterprise distributed servers. We plan to incorporate eight bytes with 64-bit self-mutation techniques, using a bit-interchange mechanism to avoid external attacks on transit data.

Author Contributions: Conceptualization, P.D.; methodology, P.D. and S.K.H.; validation, S.S.A. and M.R.; formal analysis, A.G., R.S. and A.D.; writing—original draft preparation, P.D.; writing—review and editing, M.R., A.A. and A.S.A.; supervision, S.K.H. and R.R.; funding acquisition, S.S.A. All authors have read and agreed to the published version of the manuscript.

Funding: This study was funded by the Deanship of Scientific Research, Taif University Researchers Supporting Project number (TURSP-2020/215), Taif University, Taif, Saudi Arabia.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data in this research paper will be shared upon request made to the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kriouile, H.; El Asri, B. Theoretical and Pragmatic Cases of a Rich-Variant Approach for a User-Aware Multi-Tenant SaaS Applications. *Comput. Inf. Sci.* **2019**, *12*, 1. [[CrossRef](#)]
2. Kabbedijk, J.; Bezemer, C.P.; Jansen, S.; Zaidman, A. Defining multi-tenancy: A systematic mapping study on the academic and the industrial perspective. *J. Syst. Softw.* **2015**, *100*, 139–148. [[CrossRef](#)]
3. Takahashi, T.; Blanc, G.; Kadobayashi, Y.; Fall, D.; Hazeyama, H.; Matsuo, S. Enabling secure multitenancy in cloud computing: Challenges and approaches. In Proceedings of the 2012 2nd Baltic Congress on Future Internet Communications, Vilnius, Lithuania, 25–27 April 2012; pp. 72–79. [[CrossRef](#)]
4. Kanade, S.; Manza, R. A Comprehensive Study on Multi Tenancy in SAAS Applications. *Int. J. Comput. Appl.* **2019**, *81*, 25–27. [[CrossRef](#)]
5. Pinto, V.H.S.C.; Luz, H.J.F.; Oliveira, R.R.; Souza, P.S.L.; Souza, S.R.S. A systematic mapping study on the multi-tenant architecture of SaaS systems. In Proceedings of the 28th International Conference on Software Engineering & Knowledge Engineering, San Francisco Bay, CA, USA, 1–3 July 2016; Volume 1, pp. 396–401. [[CrossRef](#)]
6. Sai, J.; Kumar, S.M.; Venkataramana, K. A Survey on Securing Cloud Service Data by Using Homomorphic Encryption. *Int. J. Comput. Sci. Eng.* **2019**, *7*, 6.
7. Saravanabhavan, C.; Anguraju, K.; Kannan, M.; Preethi, P.; Asokan, R. Ensuring Efficient Data Storage Using Fully Mature Homomorphic Encryption Technique in the Cloud Environment. *Int. J. Recent Technol. Eng.* **2019**, *8*, 4820–4832. [[CrossRef](#)]
8. Rathod, A.D.; Shah, S.A.; Jariwala, V.J. Evaluating performance of asymmetric homomorphic encryption algorithms for privacy preservation in location based services. *Int. J. Recent Technol. Eng.* **2019**, *8*, 2191–2194. [[CrossRef](#)]
9. El-yahyaoui, A.; Elkettani, M.D. Fully homomorphic encryption: State of art and comparison. *Int. J. Comput. Sci. Inf. Secur.* **2016**, *14*, 159–168. [[CrossRef](#)]
10. Ahmad, I.; Khandekar, A. Homomorphic Encryption Method Applied to Cloud Computing. *Natl. Days Netw. Secur. Syst.* **2014**, *4*, 1519–1530.
11. Prasad, S.V.S.; Kumanan, K. Homomorphic Encryption Using Enhanced BGV Encryption Scheme For Cloud Security. *Int. J. Eng. Comput. Sci.* **2018**, *7*, 23785–23789. [[CrossRef](#)]
12. Vnkv, G.; Rao, S.; Uma, G. An efficient secure message transmission in mobile ad hoc networks using enhanced homomorphic encryption scheme. *GJCST-E Netw. Web Secur.* **2013**, *13*, 1–14.
13. Jain, N. Implementation and Analysis of Homomorphic Encryption Schemes. *Int. J. Cryptogr. Inf. Secur.* **2012**, *2*, 27–44. [[CrossRef](#)]
14. Bethencourt, J.; Sahai, A.; Waters, B. Ciphertext-Policy Attribute-Based Encryption. In Proceedings of the 2007 IEEE Symposium on Security and Privacy (SP '07), Berkeley, CA, USA, 20–23 May 2007.
15. Helil, N.; Rahman, K. CP-ABE access control scheme for sensitive data set constraint with hidden access policy and constraint policy. *Secur. Commun. Netw.* **2017**, *2017*, 2713595. [[CrossRef](#)]
16. Sun, P.J. Security and privacy protection in cloud computing: Discussions and challenges. *J. Netw. Comput. Appl.* **2020**, *160*, 102642. [[CrossRef](#)]
17. Ethelbert, O.; Moghaddam, F.F.; Wieder, P.; Yahyapour, R. A JSON Token-Based Authentication and Access Management Schema for Cloud SaaS Applications. In Proceedings of the 2017 IEEE 5th International Conference on Future Internet of Things and Cloud (FiCloud), Prague, Czech Republic, 21–23 August 2017.
18. Wang, Y.; Sun, Y.; Lin, Z.; Min, J. Container-Based Performance Isolation for Multi-Tenant SaaS Applications in Micro-Service Architecture. *J. Phys. Conf. Ser.* **2020**, *1486*, 052032. [[CrossRef](#)]
19. Kanagavalli, R.; Vagdevi, S. Secured Data Storage in Cloud Using Homomorphic Encryption. *Int. J. Cloud Comput. Serv. Archit.* **2019**, *9*, 1–11. [[CrossRef](#)]
20. Chen, L.; Tong, Z.; Liu, W.; Gao, C. Non-interactive exponential homomorphic encryption algorithm. In Proceedings of the 2012 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, Sanya, China, 10–12 October 2012; Volume 2012, pp. 224–227. [[CrossRef](#)]
21. Ochei, L.C.; Bass, J.M.; Petrovski, A. Degrees of tenant isolation for cloud-hosted software services: A cross-case analysis. *J. Cloud Comput.* **2018**, *7*, 22. [[CrossRef](#)]
22. Sammeta, N.; Parthiban, L. Medical data analytics for secure multi-party-primarily based cloud computing utilizing homomorphic encryption. *J. Sci. Ind. Res.* **2021**, *80*, 692–698.
23. Awadallah, R. Verifiable Homomorphic Encrypted Computations for Cloud Computing. *Int. J. Adv. Comput. Sci. Appl.* **2021**, *12*, 10. [[CrossRef](#)]
24. Bhatia, D.; Dave, M. Partial and Fully Homomorphic Encryption Schemes for Privacy Preserving. In Proceedings of the International Conference on Advancements in Computing & Management (ICACM) 2019, Dhaka, Bangladesh, 13–14 April 2019.
25. Oladunni, T.; Sharma, S. Homomorphic Encryption and Data Security in the Cloud. *Procedia Comput. Sci.* **2019**, *79*, 175–181.
26. Al-Mashadi, H.M.; Khalaf, A.A. Hybrid homomorphic cryptosystem for secure transfer of color image on public cloud. *J. Theor. Appl. Inf. Technol.* **2018**, *96*, 6474–6486.
27. Parmar, P.V.; Padhar, S.B.; Patel, S.N.; Bhatt, N.I.; Jhaveri, R.H. Survey of Various Homomorphic Encryption algorithms and Schemes. *Int. J. Comput. Appl.* **2014**, *91*, 26–32. [[CrossRef](#)]
28. Saeed, Z.R. Improved Cloud Storage Security of Using Three Layers Cryptography Algorithms. *J. Comput. Sci.* **2018**, *16*, 34–39.

29. Rakesh, B.; Lalitha, K.; Ismail, M.; Sultana, H.P. Distributed Scheme to Authenticate Data Storage Security in Cloud Computing. *Int. J. Comput. Sci. Inf. Technol.* **2017**, *9*, 59–66. [[CrossRef](#)]
30. Dahiya, N.; Rani, S. Implementing Multilevel Data Security In Cloud Computing. *Int. J. Adv. Res.Comput. Sci.* **2017**, *8*, 146–152. [[CrossRef](#)]
31. Awasthi, U. Token Based Authentication Using Hash Key, Session and Javamail Api. *Int. J. Innov. Res. Comput. Commun. Eng.* **2017**, *5*, 12377–12384. [[CrossRef](#)]
32. Al-attab, B.S. Authentication Technique by Using USB Token in Cloud Computing. Available online: http://ici2tm.sinhgad.edu/pcproc/ICI2TM2016_P/data/IC16031.pdf (accessed on 25 December 2019).
33. Acar, A.; Aksu, H.; Uluagac, A.S.; Conti, M. A survey on homomorphic encryption schemes: Theory and implementation. *ACM Comput. Surv.* **2018**, *51*, 1–35. [[CrossRef](#)]
34. Sastry, J.K.R.; Basu, M.T. Securing multi-tenancy systems through multi DB instances and multiple databases on different physical servers. *Int. J. Electr. Comput. Eng.* **2019**, *9*, 1385. [[CrossRef](#)]
35. Hariss, K.; Chamoun, M.; Samhat, A.E. On DGHV and BGV fully homomorphic encryption schemes. In Proceedings of the 2017 1st Cyber Security in Networking Conference (CSNet), Rio de Janeiro, Brazil, 18–20 October 2017. [[CrossRef](#)]
36. Brakerski, Z.; Gentry, C.; Vaikuntanathan, V. (Leveled) fully homomorphic encryption without bootstrapping. *ACM Trans. Comput. Theor.* **2014**, *6*, 1–36. [[CrossRef](#)]
37. Gauravaram, P. Security analysis of salt password hashes. In Proceedings of the 2012 International Conference on Advanced Computer Science Applications and Technologies (ACSAT), Kuala Lumpur, Malaysia, 26–28 November 2013; pp. 25–30. [[CrossRef](#)]
38. Rangasami, K.; Vagdevi, S. Comparative study of homomorphic encryption methods for secured data operations in cloud computing. In Proceedings of the 2017 International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques (ICEECCOT), Mysuru, India, 15–16 December 2017; pp. 551–556. [[CrossRef](#)]