Performance Evaluation of RSA-based Secure Cloud Storage Protocol using OpenStack

Muhammad Faraz Hyder
Department of Software Engineering
NED University of Engineering and Technology
Karachi, Pakistan
farazh@neduet.edu.pk

Syeda Tooba
Department of Computer Science & IT
NED University of Engineering and Technology
Karachi, Pakistan
stooba1314@gmail.com

Waseemullah
Department of Computer Science and IT
NED University of Engineering and Technology
Karachi, Pakistan
waseemu@neduet.edu.pk

Abstract-In this paper, the implementation of the General Secure Cloud Storage Protocol is carried out and instantiated by a multiplicatively Homomorphic Encryption Scheme (HES). The protocol provides a system for secure storage of data over the cloud, thereby allowing the client to carry out the operational tasks on it efficiently. The work focuses on the execution of five major modules of the protocol. We also evaluate the performance of the protocol with respect to the computation cost of these modules on the basis of different security parameters and datasets by conducting a series of experiments. The cloud was built using OpenStack and the data were outsourced from the client's system to the cloud to study the security features and performance metrics when adopting the cloud environment.

Keywords-cloud computing; OpenStack; RSA; homomorphic encryption; privacy

I. INTRODUCTION

Cloud computing has become an evolving field in research and development because it can considerably lower computation cost. It has become a demanding and tempting technology that provides various services of storage, computation, and availability of data from anywhere and at any time [1]. As cloud computing is used to store a huge amount of data that reside on servers that are not physically in the reach of the users and they are transferred to and from the servers, security has therefore become an issue that obstructs the deployment of the cloud environment in an affirmative way [2-3]. The advancements in cloud computing are taking place at a very fast pace as the majority of the IT firms have either already developed IT products aligned with the cloud computing paradigm or are planning to move their infrastructure over it. This leads to a greater emphasis on improving security policies and measures. Thus, there has been a rapid emergence in the security domain of cloud computing. Storage facilities are provided by the cloud to store the data on the servers, but users are not able to verify whether their data are kept secure. Detailed overviews of the core issues concerning privacy and security in cloud computing can be seen in [4-7].

To perform operations on the encrypted data stored on the cloud requires the cloud to send data to the client. The client then needs to decrypt them and perform the required operations. Next, the data must then be encrypted again by the client before they are outsourced to the cloud. Obviously, this increases the overhead at the client and the cloud side as well and is neither a feasible nor an efficient system. It implies the need for such a system that allows the client to directly perform operations on the outsourced data without decryption. This is what we call Homomorphic Encryption Schemes (HESs) [8]. Simply encrypting the data using HESs does not cater to the demands of data security. Besides, maintaining the integrity of data sent over to the cloud, it must also allow the client to detect any forgery and recover the original data. In this paper, the implementation of the General Secure Cloud Storage Protocol instantiated by RSA-based HES has been carried out and evaluated. The cloud was built using OpenStack and data from the client's system were outsourced to the cloud. Also, the performance of the protocol was evaluated with respect to the computation cost of these modules on the basis of different security parameters and datasets by conducting a series of experiments. This work also gives an overview to understand the practicality of the RSA-SCS protocol to overcome the threats and issues posed to the data stored on the cloud.

Authors in [9], studied the execution of gradient-based algorithms that belong to a class of distributed projected algorithms. They proposed HE-based schemes that can attain accuracy by concurrently maintaining the state and coefficient of the respective member. Authors in [10], dealt with a privacy-preserving distributed big data analytics framework for cloud-based applications that performs analysis tasks on encrypted data with end-to-end data protection. The framework employs

Corresponding author: Muhammad Faraz Hyder

BGV as a fully HES. Authors in [11] reviewed HESs and also presented an implementation of the Paillier PHE using Python Library for Paillier's PHE. The extension module Gmpy2 was used for carrying out the arithmetic operations and calculations. Authors in [12] discussed the enhanced FHE with a focus on the dynamic structure of Fully HESs that are symmetric in nature. Regarding secure outsourced calculations using HE, the authors in [13] discussed the encrypted data processing protocols by applying HE over addition and proxy cryptography. Their system aimed to attain the secrecy of the user data while preserving the intermediary and final results security under the semi-honest model. To process encrypted data, non-interactive protocols were designed. Authors in [14] discussed the security of multiparty computations. Sometimes, the users of cloud infrastructure have to sustain the computation overheads although these must be outsourced.

II. RSA BASED SECURE CLOUD STORAGE PROTOCOL

To work out on the performance evaluation of the protocol, we have used RSA-based HES to instantiate the General Cloud Storage Protocol discussed in [15]. The G-SCS System Model RSA-based HES is partially homomorphic over multiplication [16].

A. The RSA-SCS Protocol

The chief aspects of the RSA-based HES are:

- KeyGen $(1^{\lambda}) \rightarrow (SK, PK)$. By considering the value of λ, this module is required to generate two prime integers p and q which must be large enough [17].
- The multiplicative homomorphic property of RSA can be computed using the technique presented in [18].
- RSA can be used to evaluate the General Cloud Storage Protocol as RSA-SCS which consists of five modules each corresponding to a module as in the G-SCS but with the different parameters and Enc( ), Dec( ). [15].

B. RSA-SCS Workflow

It is evident that to validate the integrity of the data stored on the cloud, the relationship expressed in [15] can be easily used by the client. The comprehensive approach to describe the modules and the steps the in RSA-SCS are mentioned below.

- KeyGen $(1^{\lambda}) \rightarrow (SK, PK)$. This module is required to generate two integers p and q, where both of these should be prime numbers and the number of bits in the product of p and q must be $\geq \lambda$.
- Outsource $(D; SK) \rightarrow D'$. The client needs to split data D to m data segments such as $(d_1, d_2, \ldots\ldots, d_m)$, where $d_i \in Zn^*$ (and Zn* denotes {0,1, …, n-1}) for each value of i, before outsourcing its own data,.
- Audit $(1^{\lambda}) \rightarrow \sigma$. As G-SCS protocol provides two types of auditing. The client can perform either deterministic auditing, which is comparatively simpler, or randomized auditing.
- In the former, the user spawns another key K_2 whose length of bits is λ, to compute PRF using this key value. Now the auditing query $\sigma = K_2$ and it is then sent to the cloud.
- In the case of randomized auditing, the client selects L values $(i_1, i_2, \ldots, i_L)$ from Z_{nm} (Z_{nm} denotes m numbers chosen from the set {0,1, …, n-1}) and also generates K_2, whose length of bits is λ to compute PRF $F_{K_2}(\cdot)$. Now, auditing query $\sigma = [(i_1, i_2, \ldots, i_L), K_2]$ and both of these are sent.
- Prove $(\sigma, D'; PK) \rightarrow \Gamma$. The proof generated by the cloud is also of two types depending on the type of audit query sent by the client. This is the output of the 'verify' module.

III. OPENSTACK CLOUD IMPLEMENTATION

OpenStack is one of the most popular open-source cloud ecosystems. We implemented the Openstack using 3-node architecture as depicted in Figure 1. The architecture consists of Compute, Controller, and Network nodes. These nodes are installed using the Ubuntu server 16.04. The OpenStack cloud was implemented in order to emulate the performance of the protocol in cloud environment. The data transfer and encryption for the instance running in the OpenStack cloud environment provided a realistic approach for the implementation of the protocol. Figure 2 presents the dashboard screen of the installed OpenStack Cloud.

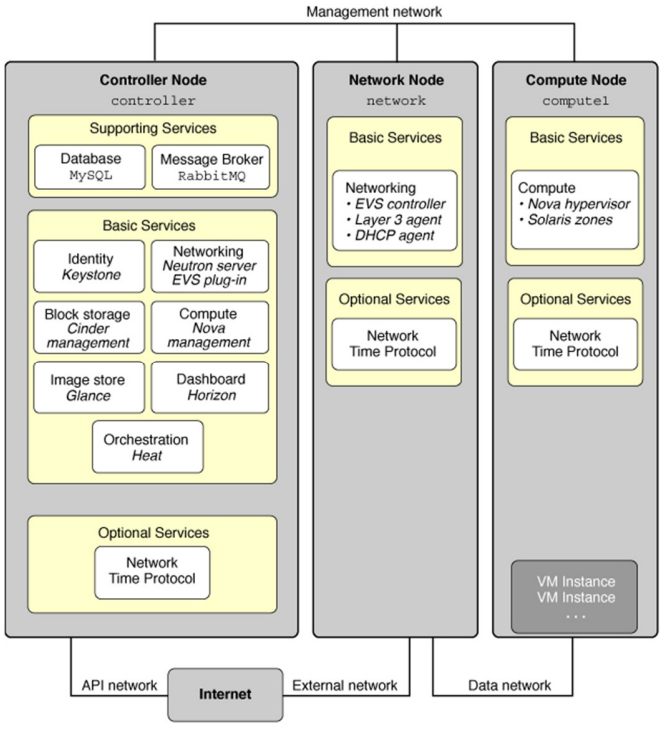

Fig. 1. OpenStack core components.

IV. EXPERIMENTAL SETUP

To perform the experiments on the RSA-SCS, we implement this protocol in java using Netbeans IDE on a system with 1.7 Intel(R) Core(TM) i5-8500 CPU running at

3.00GHz and 16GB RAM. The evaluations were carried out using 3 different values for security parameter λ, i.e. 512, 1024, and 2048 bits on two data sets that are text files of 10 and 16KB. As the security of the RSA encryption scheme requires the product of two primes to be sufficiently large [17], therefore the BigInteger datatype is used to store the values for computation of the subsequent modules. AES is used as a pseudo-random function and the keys for the PRF are generated by the client.

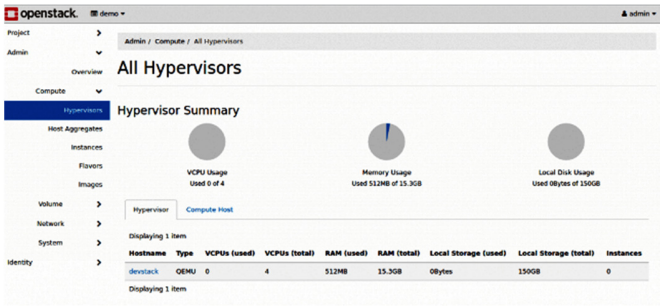

Fig. 2. OpenStack dashboard.

Each experiment was carried out 10 times and the average value was obtained as the result. To outsource the data, the Ubuntu machine was used and the instances were generated with the flavor m1.small having 2GB RAM and 20GB hard disk. The launched instances were accessed through the Putty software by using the key pair generated.

V. RESULTS AND DISCUSSION

We have implemented the cloud using OpenStack. The Ubuntu 16.04 system was used to carry out the experiment. The keystone service was used to establish the connection using the dashboard. The method involves creating new instances on the Compute node and then transferring the data over the cloud. The reason for the data values being larger in the results is that the experiment is carried out using the GUI of the instance created in the Openstack. The computation costs in seconds of the 5 modules of the RSA-SCS with respect to the 3 security parameter values when outsourcing a file of 16KB size are shown below. The values in Tables I and II signify that there is a direct correspondence between the outsourced data and the time cost of all the modules of the protocol. It can be seen from the results that running the KeyGen algorithm takes a longer time as the client generates 2 prime numbers initially and then other 3 integers to compute the pair of keys that are required for encryption and decryption. Also, the user has to generate a random number with a bit length of λ as the PRF key. The outsourcing time of KeyGen is much less than that of RSA and is linearly dependent on the size of the data, as we have carried out deterministic auditing. An audit is basically the time used to generate K_2 for the PRF as the audit query is equal to K_2. The computation cost for RSA-SCS is comparatively higher than that of RSA-SCS, since it has to parse the outsourced data and generate the values of α and β using the exponential function. Referring to Table I, the cost of RSA-SCS is as large as expected. To check whether the cloud has maintained the integrity of the data that are sent to it, the client has to carry out exponentiation and modular multiplication and then a comparison of each computed result with the corresponding value of α. Therefore, the time for this module is sufficiently large.

TABLE I. COMPUTATION COST OF file1.txt FOR VARYING BITLENGTHS

Security parameter (λ)	KeyGen (s)	Outsource (s)	Audit (s)	Prove (s)	Verify (s)
512	12.326	0.209	0.103	0.698	12.248
1024	85.788	0.780	0.793	1.587	84.952
2048	404.51	8.330	25.819	28.087	404.336

TABLE II. COMPUTATION COST OF file2.txt FOR VARYING BITLENGTHS

Security parameter (λ)	KeyGen (s)	Outsource (s)	Audit (s)	Prove (s)	Verify (s)
512	15.423	0.314	0.312	0.956	16.738
1024	102.88	1.764	0.993	3.485	104.512
2048	503.32	11.234	40.819	54.486	604.334

Figures 3 to 7 represent the comparative analysis of different parameters for 2 different file sizes, i.e. 10 and 16KB. As evident from these graphs, the time required for different parameters increases linearly with the increase in file size.

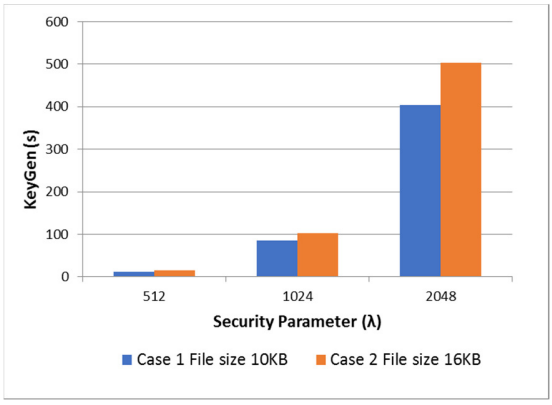

Fig. 3. Comparative analysis of KeyGen parameter for different file sizes.

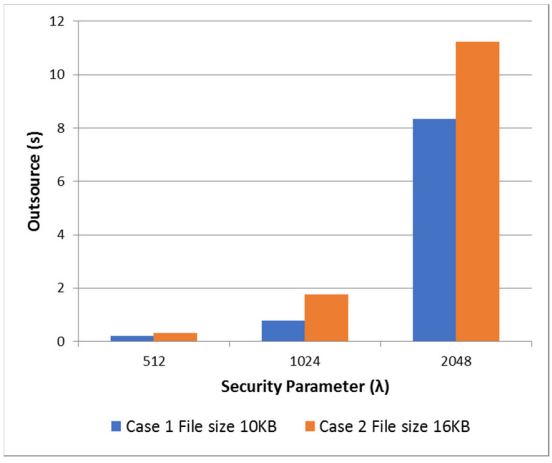

Fig. 4. Comparative analysis of Outsource parameter for different file sizes.

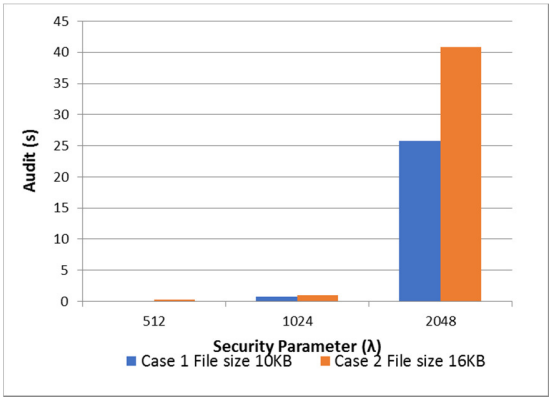

Fig. 5. Comparative analysis of Audit parameter for different file sizes.

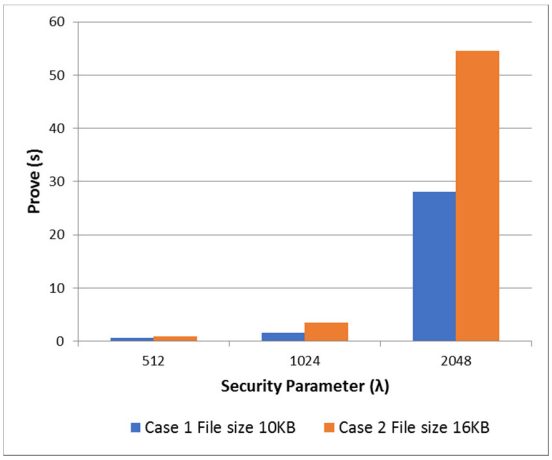

Fig. 6. Comparative analysis of Prove parameter for different file sizes.

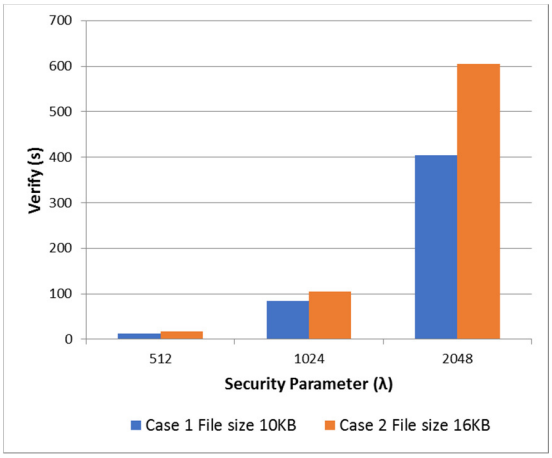

Fig. 7. Comparative analysis of Verify parameter for different file sizes.

Table III shows the comparison of the proposed approach with the existing solutions. The proposed scheme was implemented using OpenStack cloud.

VI. G-SCS IMPLEMENTATION: SECURE AGAINST RISKS

The data stored on the cloud face threats such as insider attacks, data leakage/loss, data alteration, malicious cloud providers, and lack of data recovery [19, 20]. The implemented OpenStack based G-SCS provides security against these types of risks and threats posed to the cloud data and users.

TABLE III. COMPARISION OF THE PROPOSED IMPLEMENTATION WITH EXISTING SOLUTIONS

	Implementation on open-source cloud platform	Third party auditing	Security model
Proposed	✓	✓	Standard
[9]	×	×	ROM
[11]	×	×	ROM
[14]	×	✓	Standard

A. Loss of Data Integrity

The OpenStack based Secure Cloud Storage Protocol implementation offers its clients verification of the integrity of the data stored on the cloud through the auditing mechanism, thereby sending the proof against the audit query which can be verified as mentioned above. The cloud cannot cheat its client since a minor change in data results in a proof which when verified by the client, leads to output 0 rather than 1.

B. Malicious Cloud Provider

This is one of the major threats to cloud computing. It can include a malicious insider/employee or the provider itself [21]. The implemented GSCS-RSA outsource initially requires a pseudorandom function which must be secure enough so that the overall protocol becomes secure. As discussed in [15], if the cloud provider is malicious, it generates more than one (legitimate) proofs. Even then, the number of unknown quantities is greater than the total number of equations thus generated and hence the malicious cloud is unable to discover the secret key because of the PRF being used.

C. Lack of Data Recovery

The data recovery algorithm provides an efficient way to reconstruct the data in case of data loss. For this, the client needs to generate a special audit query for which the cloud sends the proof. If this proof is verified by the client, he then computes $\mathrm{Enc}(d_i)$ by calculating $\beta = \mathrm{Enc}(d_i)^{e_i} \bmod t$. Finally the outsourced data $(d_1, d_2, \ldots\ldots, d_m)$ can easily be recovered from $\mathrm{Enc}(d_i)$.

VII. CONCLUSION AND FUTURE WORK

In this paper, we have carried out an evaluation on the performance metrics and studied the security properties of the General Sloud Storage protocol under various circumstances that can occur in any HES. We have instantiated this protocol by a HES that is multiplicatively homomorphic, i.e. RSA, and thus assessed the computation cost of RSA-SCS using various parameter values and data sets in a series of experiments. The carried out analysis left us with the conclusion that although the evaluated protocol is secure and efficient, its computation costs can be lowered. As the experiments were conducted on varying bitlengths of the security parameter, it was found that, the higher the value of λ, the greater the security achieved, but the computation cost increases. On the other hand, a 1024 bitlength can help us attain more or less the same security but with a greater impact on the computation cost. The computation costs of the 5 modules with respect to different values of security

parameters are found by applying this algorithm on two datasets which led us to the optimization of the RSA-SCS protocol.

There is always a door towards improvement, therefore we can further take steps to make this protocol operate more securely while maintaining its efficiency. Moreover, since RSA is a partially HES, we can further extend the protocol so that it is instantiated by an encryption scheme that is fully homomorphic. Although fully HESs are required in real-life applications and cloud infrastructure, their cost, whether that of computation or performance, is high. Therefore, as future work, we can consider this aspect and make progress in this regard that can lead us to achieve a secure as well as lightweight scheme. In the future, we will also implement the proposed scheme in a federated cloud environment.

REFERENCES

[1] M. A. Shahid, N. Islam, M. M. Alam, M. S. Mazliham, and S. Musa, "Towards Resilient Method: An exhaustive survey of fault tolerance methods in the cloud computing environment," *Computer Science Review*, vol. 40, May 2021, Art. no. 100398, https://doi.org/10.1016/j.cosrev.2021.100398.

[2] M. Ali, N. Q. Soomro, H. Ali, A. Awan, and M. Kirmani, "Distributed File Sharing and Retrieval Model for Cloud Virtual Environment," *Engineering, Technology & Applied Science Research*, vol. 9, no. 2, pp. 4062–4065, Apr. 2019, https://doi.org/10.48084/etasr.2662.

[3] M. Ramzan, M. S. Farooq, A. Zamir, W. Akhtar, M. Ilyas, and H. U. Khan, "An Analysis of Issues for Adoption of Cloud Computing in Telecom Industries," *Engineering, Technology & Applied Science Research*, vol. 8, no. 4, pp. 3157–3161, Aug. 2018, https://doi.org/10.48084/etasr.2101.

[4] S. Pearson, "Taking account of privacy when designing cloud computing services," in *2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing*, Vancouver, Canada, May 2009, pp. 44–52, https://doi.org/10.1109/CLOUD.2009.5071532.

[5] H. Tabrizchi and M. Kuchaki Rafsanjani, "A survey on security challenges in cloud computing: issues, threats, and solutions," *The Journal of Supercomputing*, vol. 76, no. 12, pp. 9493–9532, Dec. 2020, https://doi.org/10.1007/s11227-020-03213-1.

[6] P. Sun, "Security and privacy protection in cloud computing: Discussions and challenges," *Journal of Network and Computer Applications*, vol. 160, Jun. 2020, Art. no. 102642, https://doi.org/10.1016/j.jnca.2020.102642.

[7] P. Yang, N. Xiong, and J. Ren, "Data Security and Privacy Protection for Cloud Storage: A Survey," *IEEE Access*, vol. 8, pp. 131723–131740, 2020, https://doi.org/10.1109/ACCESS.2020.3009876.

[8] A. Acar, H. Aksu, A. S. Uluagac, and M. Conti, "A Survey on Homomorphic Encryption Schemes: Theory and Implementation," *ACM Computing Surveys*, vol. 51, no. 4, pp. 79:1-79:35, Jul. 2018, https://doi.org/10.1145/3214303.

[9] Y. Lu and M. Zhu, "Privacy preserving distributed optimization using homomorphic encryption," *Automatica*, vol. 96, pp. 314–325, Oct. 2018, https://doi.org/10.1016/j.automatica.2018.07.005.

[10] A. Alabdulatif, I. Khalil, and X. Yi, "Towards secure big data analytic for cloud-enabled applications with fully homomorphic encryption," *Journal of Parallel and Distributed Computing*, vol. 137, pp. 192–204, Mar. 2020, https://doi.org/10.1016/j.jpdc.2019.10.008.

[11] M. Nassar, A. Erradi, and Q. M. Malluhi, "Paillier's encryption: Implementation and cloud applications," in *2015 International Conference on Applied Research in Computer Science and Engineering (ICAR)*, Beiriut, Lebanon, Oct. 2015, https://doi.org/10.1109/ARCSE.2015.7338149.

[12] K. Hariss, H. Noura, and A. E. Samhat, "Fully Enhanced Homomorphic Encryption algorithm of MORE approach for real world applications," *Journal of Information Security and Applications*, vol. 34, pp. 233–242, Jun. 2017, https://doi.org/10.1016/j.jisa.2017.02.001.

[13] Q. Wang, D. Zhou, and Y. Li, "Secure outsourced calculations with homomorphic encryption," *Advanced Computing: An International Journal*, vol. 9, no. 6, pp. 01–14, Nov. 2018, https://doi.org/10.5121/acij.2018.9601.

[14] L. Jiang, Y. Cao, C. Yuan, X. Sun, and X. Zhu, "An effective comparison protocol over encrypted data in cloud computing," *Journal of Information Security and Applications*, vol. 48, Oct. 2019, Art. no. 102367, https://doi.org/10.1016/j.jisa.2019.102367.

[15] J. Zhang, Y. Yang, Y. Chen, J. Chen, and Q. Zhang, "A general framework to design secure cloud storage protocol using homomorphic encryption scheme," *Computer Networks*, vol. 129, pp. 37–50, Dec. 2017, https://doi.org/10.1016/j.comnet.2017.08.019.

[16] R. L. Rivest, L. Adleman, and M. L. Dertouzos, "On Data Banks and Privacy Homomorphisms," in *Foundations of Secure Computation*, Academia Press, 1978.

[17] J. Buhler, P. L. Montgomery, R. Robson, and R. Ruby, *Technical report implementing the number field sieve*. Corvallis, OR, USA: Oregon State University, 1994.

[18] O. Goldreich, S. Goldwasser, and S. Micali, "How to construct random functions," *Journal of the ACM*, vol. 33, no. 4, pp. 792–807, Aug. 1986, https://doi.org/10.1145/6490.6503.

[19] N. vurukonda and B. T. Rao, "A Study on Data Storage Security Issues in Cloud Computing," *Procedia Computer Science*, vol. 92, pp. 128–135, Jan. 2016, https://doi.org/10.1016/j.procs.2016.07.335.

[20] A. J. Duncan, S. Creese, and M. Goldsmith, "Insider Attacks in Cloud Computing," in *2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications*, Liverpool, UK, Jun. 2012, pp. 857–862, https://doi.org/10.1109/TrustCom.2012.188.

[21] Md. T. Khorshed, A. B. M. S. Ali, and S. A. Wasimi, "A survey on gaps, threat remediation challenges and some thoughts for proactive attack detection in cloud computing," *Future Generation Computer Systems*, vol. 28, no. 6, pp. 833–851, Jun. 2012, https://doi.org/10.1016/j.future.2012.01.006.