Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 7, Number 4, October 2023
DOI : https://doi.org/10.33395/sinkron.v8i4.12819
e-ISSN : 2541-2019
p-ISSN : 2541-044X

Performance Comparison between Signature Cryptography: A Case Study on SNAP Indonesia

Moehammad Ramadhoni[1)*], Handri Santoso[2)]
[1,2)]Universitas Pradita, Indonesia,
[1)]moehammad.ramadhoni@student.pradita.ac.id, [2)]handri.santoso@pradita.ac.id

Submitted : Jul 31, 2023 | **Accepted** : Aug 17, 2023 | **Published** : Oct 1, 2023

Abstract: SNAP (Standar Nasional OPEN API Pembayaran) was submitted by several sub-working groups formed jointly by ASPI and the Bank of Indonesia for encouraging digital transformation in the banking industry. In the document Pedoman Tata Kelola (Bank of Indonesia, n.d.), there is the use cryptographic algorithms that are used as validation for third parties to use the Open API. The algorithms used in the document are HMAC and RSA. However, there are other algorithms that can be used as a form of validation, such as ECC and ZK-SNARK. ECC uses an elliptic curve as a standard cryptography calculation which can use shorter keys than RSA. On the other hand, ZK-SNARK uses a pairing-based elliptic curve which makes verification calculations simpler. The method used as authentication in SNAP is the third party will send the signature in the API header along with the sent API payload. The signature describes the body payload, the endpoint URL that was called by the third party, and the time when the API call was made, so the signature will change all the time. In this research, the performance of the four cryptographic algorithms is compared based on SNAP method. The performance we compare is overall speed of process when creating the signature and verifying it. The result is that HMAC is the most efficient algorithm, but for financial data, it is better to use ECC which uses asymmetric keys and is faster than RSA contained in the SNAP document, especially when 256 bits security level that ECC could be 10 times faster then RSA.

Keywords: cryptographic, ECC, HMAC, performance, RSA, SNAP, ZK-SNARK,

INTRODUCTION

On 16 August 2021, Bank Indonesia verified a document regarding the use of an Open API for payments in Indonesia named SNAP (Standar Nasional OPEN API Pembayaran). This document was initiated by many parties who are members of a sub-working group called ASPI (Asosiasi Sistem Pembayaran Indonesia). The document already contains a chapter on standard encryption components that are used as part of verifying every API call made by a third party.

This chapter explains the procedures for using signatures for the verification process, such as using public and private keys, using encoding authentication in each header, and using signatures in headers. The use of data encryption is intended as an additional security for API calls so that third parties cannot call any API with the same signature when calling other APIs.

*Moehammad Ramadhoni

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 7, Number 4, October 2023 e-ISSN : 2541-2019
DOI : https://doi.org/10.33395/sinkron.v8i4.12819 p-ISSN : 2541-044X

There are two uses of signatures in SNAP documentation, namely symmetric and asymmetric signatures. Symmetric signature uses the HMAC-SHA512 algorithm. On the other hand, an asymmetric signature uses the SHA256-with-RSA algorithm. Both of these signatures have an adequate good level of security and are difficult to be attacked by the man in the middle.

The difference between the two signatures is that RSA requires a private and public key that is generated at the beginning as a trusted key that is stored by both parties, while HMAC does not require a trusted key, because the process of forming a signature only requires a secret key that is known by both parties. The SHA256-with-RSA technique also requires a secret key for the SHA process in the algorithm.

Other authentication processes are ECC and ZK-SNARK. Both use an elliptic curve which has a high level of security with a shorter key length. With a shorter key length, the verification process becomes faster and more efficient. This elliptic curve algorithm is used in web3 applications and blockchain applications which are widely used today.

In this research, we will compare the signature algorithm processes on SNAP, namely HMAC-SHA512 and SHA256-with-RSA with ZK-SNARK. ZK-SNARK is an authentication algorithm using pair elliptic curve cryptography which can reduce the proof for verification but still with a high level of security. The results of this comparison will lead to increased performance but still with the same level of security.

LITERATURE REVIEW

RSA has been deemed as a secure and trustworthy algorithm among all asymmetric algorithms which have been proposed up to now. In fact, the RSA algorithm is a compatible asymmetric cipher, since it applies a key with various length. In this algorithm security can be assured at the expense of speed. The typical length of RSA keys are 512- 2048 bits. Rivest et al invented RSA algorithm in 1978 (Rivest, 1978).

Considerable cryptanalysis has approved RSA as a reliable algorithm over the years. It demonstrates that this algorithm has remarkable amount of reliability. Difficulty of factoring large numbers acts as a core component of RSA's security. The efficiency of RSA would be ruined if it was possible to find a simple method for factoring these large numbers.

MAC algorithms are keyed hash functions that allow to verify whether a transmitted message has been altered. In order to use a MAC algorithm in computer networks, a secret key should be first distributed to the authorized entities. HMAC, which was designed by Bellare, Canetti and Krawczyk, is a standardized hash-based MAC algorithm that is widely used as a MAC algorithm and as a pseudorandom function generator (Bellare, 1996). HMAC takes a message of an arbitrary bit-length and hashes it with one secret key.

HMAC is proved to be a pseudorandom function under the assumption that the compression function of the underlying hash function is a pseudorandom function (Bellare, 2006) (note that the security proof of pseudorandomness provides the MAC security (Bellare, 2000)). However, this does not guarantee the security of HMAC if it is instantiated with a specific cryptographic hash function such as MD5 or SHA-1.

The utilization of elliptic curves in cryptography has been proposed for the first time by Koblitz and Victor Miller individually in mid 1980s (Koblitz, 1987). ECC is known as a sort of PKC which is built upon algebraic structure of elliptic curve over finite fields. ECC use computation which known as Elliptic Curve Discrete Logarithm Problem (ECDLP). This algorithm can only be resolved in exponential time causing ECC became a promising branch of public-key cryptography which offers

*Moehammad Ramadhoni

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 7, Number 4, October 2023 e-ISSN : 2541-2019
DOI : https://doi.org/10.33395/sinkron.v8i4.12819 p-ISSN : 2541-044X

similar security to other ”traditional” DLP- based schemes, with smaller key sizes and memory requirements, e.g., 160 bits instead of 1024 bits.

The National Institute of Standards and Technology (NIST) is a federal non-standards agency within the U.S. Department of Commerce Administration of Technology. NIST provides specifications for ECC that are considered safe for application in cryptography. NIST recommends elliptic curves in binary-fields with values 2^{163},2^{233}, 2^{248}, 2^{409}, and 2^{571} (Hankerson, 2000).

The basic concept of zero-knowledge proof is that the prover exchanges messages with the verifier, where the prover tries to convince the verifier that the prover knows something without having to tell the verifier something. Unlike interactive proofs, no-interactive proofs only require one interaction between participants. Prover sends confidential information to a special algorithm that can calculate zero-knowledge proofs. The proof is then sent to the verifier, which will check the confidential information using another algorithm. Non-interactive zero-knowledge proof reduces communication between the prover and verifier, making ZKP more efficient.

Groth (Groth, 2016) builds on NIZK's argument to satisfy an arithmetic circuit where the proof consists of only 3 groups of elements. Due to their small size, proofs are also easier to verify. The verifier only needs to calculate an exponential number proportional to the statement size and check the equation for the single pairing product, which has only 3 pairs

METHOD

Based on the SNAP document from Bank Indonesia, Open API is used by third parties to access services from banks or similar parties. To be able to access these services, an authentication process is used using the header in the API call. The header contains signatures that are only expected to be known by third parties and API owners so that it is expected to minimize the possibility of unauthenticated parties being able to access and enjoy the service.

In this section, we will compare the performance of the two algorithms used in SNAP, namely RSA and HMAC, with the cryptographic algorithms that are currently widely used, namely ECC and ZK-SNARK. All four algorithms will be performed at the middleware level, it is the schema where implement before the request process is performed. The process and flow for creating a signature follow the directions in the SNAP documen t so that system development also follows the document.

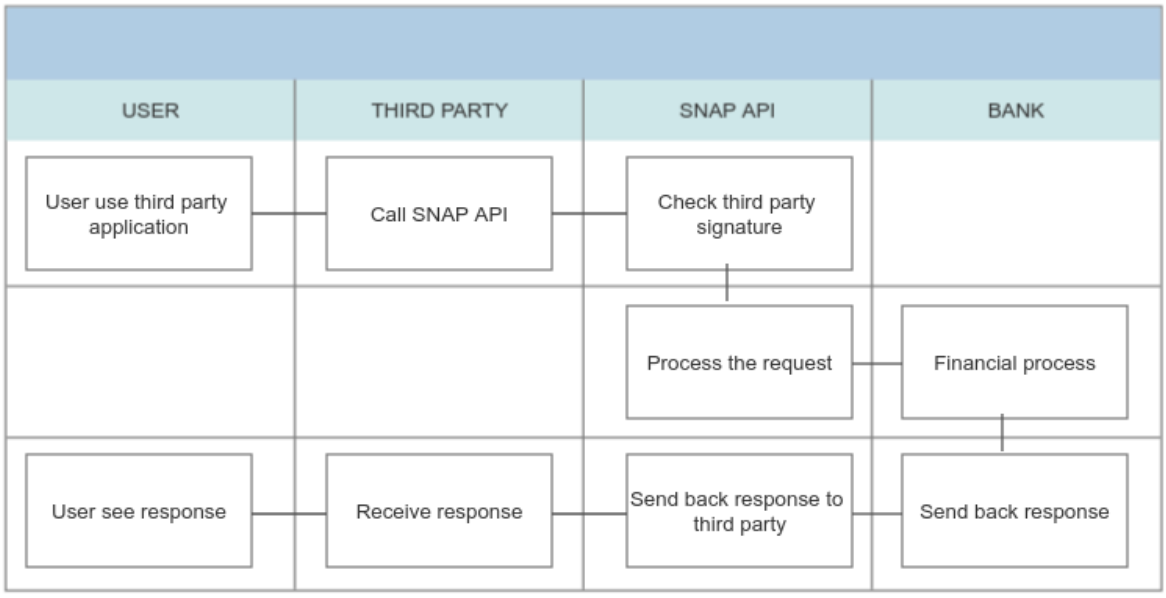

Fig 1: SNAP API process

*Moehammad Ramadhoni

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 7, Number 4, October 2023
DOI : https://doi.org/10.33395/sinkron.v8i4.12819
e-ISSN : 2541-2019
p-ISSN : 2541-044X

To develop the system, the Golang programming language was used. For making the API, the echo library is used (Labstack, 2021) which has been used by many developers as the main library for API development in the Golang language. To develop the RSA, HMAC, and ECC algorithms, the crypto library (default library in Golang) is used, while ZK-SNARK uses the gnark library (Gautam, n.d., 2023).

This research uses the same security level for each cryptographic algorithm used so that a cost ratio is obtained for each time of creation and signature verification for each algorithm. Below is the algorithm used for each cryptography.

Table 1: Key size ratio between HMAC, RSA, ECC, and ZK-SNARK

Key Size				Security Level (bits)	Key Size Ratio
HMAC	**RSA**	**ECC**	**ZK-SNARK**		
256*	3072	256	256	128	1:12:1:1
512*	15360	521	388**	256	1.3:40:1.3:1

**key and message must satisfied standard size so security level could be achieve*

**implementation of *pairing-ECC nowday max uses curve BW6-761 that could create 388 bits key with security level 192 bits.[14]*

HMAC and ECC have the same key size ratio, whereas the higher the desired security level, the higher the key size required by RSA to achieve that security. The largest current implementation of ZK-SNARK uses a key size of 388 bits. The size of the key will affect the output length of the signature. Based on Table 1, at a security level of 128 bits, RSA will have an output signature 12 times longer than HMAC and ECC, while at a security bit of 256, RSA will produce an output signature length 30 times longer than HMAC and ECC.

In the HMAC algorithm, it takes a key length equal to the number of algorithms used so that the expected security level can be achieved. Otherwise, the security level will only be the same as the bit length of the key, for example in the HMAC-SHA256 algorithm, the key length used must be 256 bits so that the security level can be maintained with a minimum message half of 256 bits.

The ZK-SNARK algorithm requires a circuit constraint before using the ZK-SNARK algorithm. The circuit is needed to verify whether the prover knows the circuit and can perform calculations based on the agreed circuit. In this research, each security level ZK-SNARK will use 2 different circuits, the first EDDSA with 256 bits and the second with 512 bits.

RESULT

Load tests are used to obtain the performance of an application by calling the endpoint of the application many times, serially or parallel (concurrent). In this research, a load test is carried out using the Jmeter application. To get good results, we must isolate the machine used for the load test and each load test must have the same environment so that stable results will be obtained for each function measured.

*Moehammad Ramadhoni

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 7, Number 4, October 2023
DOI : https://doi.org/10.33395/sinkron.v8i4.12819
e-ISSN : 2541-2019
p-ISSN : 2541-044X

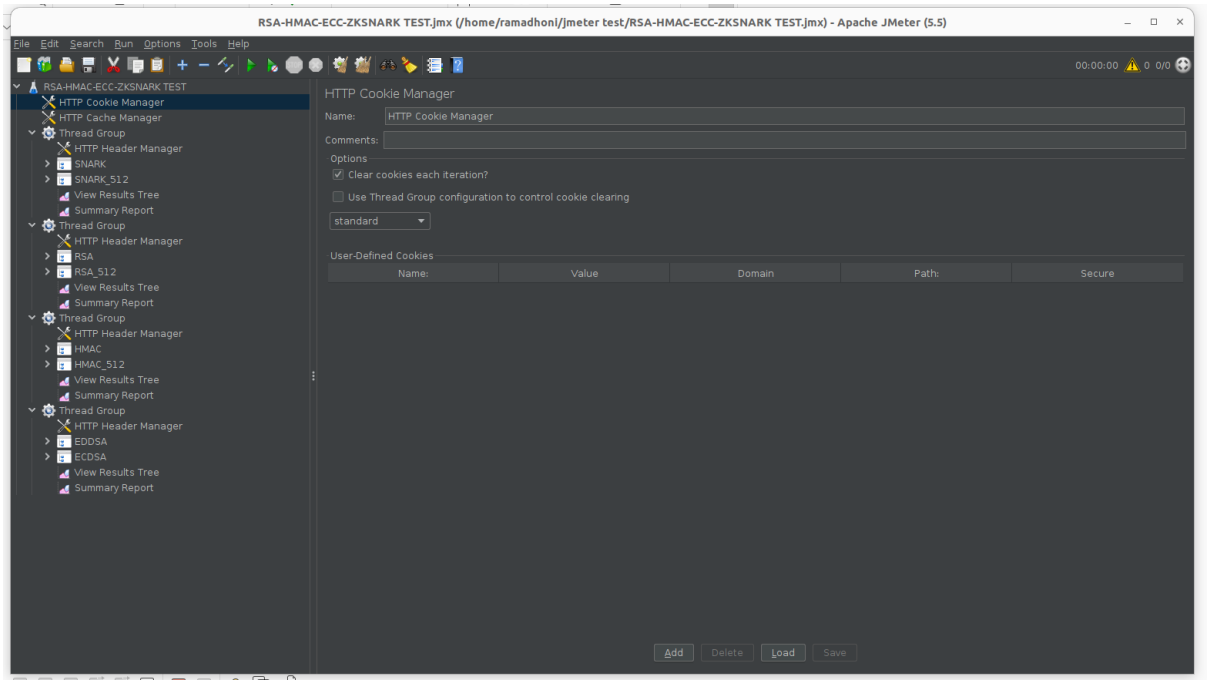

Fig 2: Application interface Jmeter

Table 2: Load test result

Label	Average (ms)	Min (ms)	Max (ms)	Std. Dev.	Throughput (KB/sec)	Received KB/sec	Sent KB/sec	Avg. Bytes
Create_HMAC_Token	1	1	2	0.49	5.1	1.38	3.22	277
Verify_HMAC_Token	1	1	2	0.37	5.1	2.14	2.96	430
Create_HMAC_512_Token	0	0	5	0.78	5.1	1.6	3.28	322
Verify_HMAC_512_Token	1	0	2	0.45	5.1	2.14	3.2	430
Create_RSA_Token	8	6	16	2.48	5.1	3.72	3.21	746
Verify_RSA_Token	1	0	2	0.36	5.1	2.14	5.29	430
Create_RSA_512_Token	477	393	583	33.69	4.91	13.48	3.14	2813
Verify_RSA_512_Token	4	3	9	1.17	5.1	2.14	15.51	430
Create_EDDSA_Token	1	1	6	0.7	5.1	1.6	3.24	322
Verify_EDDSA_Token	1	1	2	0.49	5.1	2.14	3.19	430

*Moehammad Ramadhoni

e-ISSN : 2541-2019
p-ISSN : 2541-044X

Create_ECDSA_Token	19	8	61	6.98	5.09	3.33	3.23	668.7
Verify_ECDSA_Token	23	19	31	2.72	5.11	2.15	4.92	430
Create_SNARK_Token	1718	1243	6538	952.74	0.04	0.03	0.03	785
Verify_SNARK_Token	69	53	181	22.42	0.02	0.01	0.03	430
Create_SNARK_512_Token	12387	10111	24405	2910.76	0.02	0.02	0.02	1041
Verify_SNARK_512_Token	132	94	548	83.54	0.02	0.01	0.03	430

The table above describes the time allocation, throughput, and data sent for each function that is executed in parallel. From the table above, the time difference between functions is greatest when generating tokens in the ZK-SNARK process. This happen because the process when creating tokens on ZK-SNARK is a high-computation process. After all, it is hoped that this computation can reduce verification time. As can be seen in the table, even though the token creation process takes an average of 12 seconds, verification only takes an average of around 132ms.

It can also be seen in the table that the process of creating and verifying signatures in ECC has a low time. The difference is felt in the creation and verification of 512 bits signatures, RSA has token generation up to 583ms, while ECDSA only 61ms. However, the RSA process has a faster verification time than the ECDSA process.

The process in the table illustrates that the most efficient signature generation is using HMAC with a maximum of 2 ms for 256 bits and 5 ms for 512 bits. This is understandable because of all signature processes, only HMAC uses a symmetric key so that the creation and verification techniques are quite easy, namely by comparing the received signature with the signature reproduced by the server.

DISCUSSIONS

From the load test results, we can see that HMAC is the most efficient signature that can be used as an authentication process between third parties and API providers in SNAP. However, HMAC uses a symmetric key, which means that the same key is used between the message sender and the message recipient. This can happen by the way the sender of the message has informed the recipient of the message the key used or the sender of the message while sending the key to the recipient of the message.

This can result in the key being known by a third party and a third party being able to decrypt the data sent, or a third party being able to encrypt it and act as if it were the sender of the message. In addition, there is a greater possibility of guessing keys compared to using asymmetric keys (Lenstra, 2001). That is why HMAC is more widely used on big data because the creation and verification process is fast, but the data used is not critical.

For asymmetric keys, the most efficient use is to use EDDSA and ECDSA. RSA has a fairly good verification process time even when using a 256 bits security level. However, the signature generation time is quite high because it uses a very long key length compared to ECC. In SNAP, signature generation is still used using RSA because RSA is more familiar in Indonesia than ECC.

On the other hand, ZK-SNARK has quite poor performance when it comes to the proof creation process compared to other schemes, reaching 2 seconds for 128 bits security and 20 seconds for 256 bits

*Moehammad Ramadhoni

security. However, making proof using ZK-SNARK is the safest process at the moment for non-interactive processes because the prover doesn't need to send how to make the proof and the verifier also doesn't need to know how the proof was made but can still find out whether the data sent is correct. No. This can make it difficult for third parties to dismantle the contents of the signature. Therefore, ZK-SNARK is used in web3 applications, where proofing can be done outside the network and the data sent is very private and no one should know, including the verifier.

CONCLUSION

From the research results above, we recommend that SNAP use the ECC cryptographic algorithm, namely EDDSA or ECDSA because it is faster and only requires shorter keys with the same level of security as RSA. Also SNAP should consider using the ECC algorithm as a cryptographic function to save data in the financial industry in the future since the data is really important and should be having some additional security.

REFERENCES

Aranha, D. F., Housni, Y. E., & Guillevic, A. (2022). A survey of elliptic curves for proof systems. Cryptology ePrint Archive, Paper 2022/586. https://eprint.iacr.org/2022/586

Bafandehkar, M., Yasin, S. M., Mahmod, R., & Hanapi, Z. M. (2013). *Comparison of ECC and RSA algorithm in resource constrained devices*. https://doi.org/10.1109/icitcs.2013.6717816.

Bank of Indonesia (n.d) Pedoman Tata Kelola SNAP. Retrieved May 01, 2023, from https://bi.go.id/id/layanan/Standar/SNAP/Documents/SNAP_Pedoman_Tata_Kelola.pdf

Bank of Indonesia (n.d) Standar Data Spesifikasi Teknis SNAP. Retrieved May 01, 2023, from https://apidevportal.bi.go.id/snap/docs/standar-data-spesifikasi-teknis

Bank of Indonesia (n.d) Standar Teknis Keamanan SNAP. Retrieved May 01, 2023, from https://apidevportal.bi.go.id/snap/docs/standar-teknis-keamanan

Bin Uzayr, S. (2022a). *Mastering golang: A beginner's guide* (1st ed.). CRC Press. https://doi.org/10.1201/9781003310457

Bin Uzayr, S. (2022b). *Golang: The ultimate guide* (1st ed.). CRC Press. https://doi.org/10.1201/9781003309055

Buterik, Vitalik. (2021) *An approximate introduction to how zk-SNARKs are possible*. Retrieved May 06, 2023, from https://vitalik.ca/general/2021/01/26/snarks.html

Dymora, P., & Paszkiewicz, A. (2020). Performance analysis of selected programming languages in the context of supporting decision-making processes for industry 4.0. *Applied Sciences (Switzerland)*, *10*(23), 1–17. https://doi.org/10.3390/app10238521

Effendy, F., Taufik, & Adhilaksono, B. (2019). Performance Comparison of Web Backend and Database: A Case Study of Node.JS, Golang and MySQL, Mongo DB. *Recent Advances in Computer Science and Communications*, *14*(6), 1955–1961. https://doi.org/10.2174/2666255813666191219104133

El Housni, Y., & Guillevic, A. (2022). Families of snark-friendly 2-chains of elliptic curves. In O. Dunkelman & S. Dziembowski (Eds.), *Advances in Cryptology – EUROCRYPT 2022* (Vol. 13276, pp. 367–396). Springer International Publishing. https://doi.org/10.1007/978-3-031-07085-3_13

Ethereum.Org .(n.d). *Zero-knowledge proofs*. Retrieved May 06, 2023, from https://ethereum.org

Gautam Botrel, Thomas Piellard, Youssef El Housni, Ivo Kubjas and Arya Tabaie. Gnark. 2023. Retrieved from https://github.com/ConsenSys/gnark

Gong, Y., Jin, Y., Li, Y., Liu, Z., & Zhu, Z. (2022). Analysis and comparison of the main zero-knowledge proof scheme. In *Proceedings - 2022 International Conference on Big Data, Information and Computer Network, BDICN 2022* (pp. 366–372). Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/BDICN55575.2022.00074

Groth, J. (2016). On the size of pairing-based non-interactive arguments. In M. Fischlin & J.-S. Coron (Eds.), *Advances in Cryptology – EUROCRYPT 2016* (Vol. 9666, pp. 305–326). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-49896-5_11

*Moehammad Ramadhoni

Harjito, B., Tyas, H. N., Suryani, E., & Wardani, D. W. (2022). Comparative analysis of rsa and ntru algorithms and implementation in the cloud. *International Journal of Advanced Computer Science and Applications (IJACSA)*, *13*(3). https://doi.org/10.14569/IJACSA.2022.0130321

Harjoseputro, Y., Albertus Ari Kristanto, & Joseph Eric Samodra. (2020). Golang and NSG Implementation in REST API Based Third-Party Sandbox System. *Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi)*, *4*(4), 745–750. https://doi.org/10.29207/resti.v4i4.2218

Housni, Y. E., & Guillevic, A. (2021). Families of SNARK-friendly 2-chains of elliptic curves. Cryptology ePrint Archive, Paper 2021/1359. https://doi.org/10.1007/978-3-031-07085-3_13

Husufa, N., & Prihandi, I. (2022). Optimizing JMeter on performance testing using the bulk data method. *Journal of Information Systems and Informatics, 4(2)*, 205–215. https://doi.org/10.51519/journalisi.v4i2.244

K. Moriarty, B. Kaliski, J. Jonsson, and A. Rusch. PKCS #1: RSA Cryptography Specifications Version 2.2. RFC 8017, 2016. https://datatracker.ietf.org/doc/html/rfc8017.

Konkin, A., & Zapechnikov, S. (2023). Zero knowledge proof and ZK-SNARK for private blockchains. *Journal of Computer Virology and Hacking Techniques*. https://doi.org/10.1007/s11416-023-00466-1

J, G., & Koppu, S. (2022). An empirical study to demonstrate that EdDSA can be used as a performance improvement alternative to ECDSA in Blockchain and IoT. *Informatica*, *46*(2). https://doi.org/10.31449/inf.v46i2.3807

J, R., N, E. E., & Asokan, N. (2022). Implementation and performance analysis of elliptic curve cryptography using an efficient multiplier. *JOURNAL OF SEMICONDUCTOR TECHNOLOGY AND SCIENCE*, *22*(2), 53–60. https://doi.org/10.5573/JSTS.2022.22.2.53

LabStack.(n.d). Echo. 2021. Retrieved from https://github.com/labstack/echo.

Lenstra, A. K., & Verheul, E. R. (2001). *Selecting Cryptographic Key Sizes. Journal of Cryptology, 14(4), 255–293.* doi:10.1007/s00145-001-0009-4

Li, W. H., Zhang, Z. Y., Zhou, Z. B., & Deng, Y. (2022, July 1). An Overview on Succinct Non-interactive Zero-knowledge Proofs. *Journal of Cryptologic Research*. Chinese Association for Cryptologic Research. https://doi.org/10.13868/j.cnki.jcr.000525

National Institute of Standards and Technology. Federal Information Processing Standards FIPS PUB 186-4: Digital Signature Standard (DSS), 2013. https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf

National Institute of Standards and Technology. Federal Information Processing Standards FIPS PUB 180-4: Secure Hash Standard, 2015. https://nvlpubs.nist.gov/nistpubs/ FIPS/NIST.FIPS.180-4.pdf.

Ogunleye, G. O., & Akinsanya, S. E. (2022). Elliptic curve cryptography performance evaluation for securing multi-factor systems in a cloud computing environment. *Iraqi Journal of Science*, 3212–3224. https://doi.org/10.24996/ijs.2022.63.7.40

Setty, S. (2020). Spartan: Efficient and general-purpose zksnarks without trusted setup. In D. Micciancio & T. Ristenpart (Eds.), *Advances in Cryptology – CRYPTO 2020* (Vol. 12172, pp. 704–737). Springer International Publishing. https://doi.org/10.1007/978-3-030-56877-1_25

Singh, S. R., Khan, A. K., & Singh, S. R. (2016). Performance evaluation of rsa and elliptic curve cryptography. *2016 2nd International Conference on Contemporary Computing and Informatics (IC3I)*, 302–306. https://doi.org/10.1109/IC3I.2016.7917979

Tyagi, S., & Kathuria, M. (2022). Role of Zero-Knowledge Proof in Blockchain Security. In *2022 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing, COM-IT-CON 2022* (pp. 738–743). Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/COM-IT-CON54601.2022.9850714

*Moehammad Ramadhoni

Ullah, S., Zheng, J., Din, N., Hussain, M. T., Ullah, F., & Yousaf, M. (2023). Elliptic Curve Cryptography; Applications, challenges, recent advances, and future trends: A comprehensive survey. *Computer Science Review*, *47*, 100530. https://doi.org/10.1016/j.cosrev.2022.100530

Yousif, S. F. (2023). Performance comparison between rsa and el-gamal algorithms for speech data encryption and decryption. *Diyala Journal of Engineering Sciences*, 123–137. https://doi.org/10.24237/djes.2023.16112

Zahan, A., Hossain, Md. S., Rahman, Z., & Shezan, Sk. A. (2020). Smart home IoT use case with elliptic curve based digital signature: An evaluation on security and performance analysis. *International Journal of Advanced Technology and Engineering Exploration*, *7*(62), 11–19. https://doi.org/10.19101/IJATEE.2019.650070

*Moehammad Ramadhoni