# Analysis and Simulation of Digital Identity Protection

## Using Blockchain-based Database, ECIES, EdDSA, and Keccak Implementations as Alternative Solution for Cyber Threats

Christopher Febrian Nugraha / 18221115 (*Author*)
Program Studi Sistem dan Teknologi Informasi
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
cfnugraha823@gmail.com

***Abstract*—In a rapidly growing digital era such as now, one must have a digital identity that works exactly like real-life identity. However, due to its fairly new nature, digital identities are vulnerable to theft and leaks caused by faulty system or even an adversary. This paper proposes a novel and secure digital identity protection system to combat the escalating challenges posed by such cyber threats. The system leverages blockchain technology to establish a secure and tamper-proof database for storing user identities. To guarantee data security principles such as confidentiality, integrity, authenticity, and non-repudiation of these stored identities, the system incorporates a combination of robust cryptographic algorithms. Elliptic Curve Integrated Encryption Scheme (ECIES) safeguards the privacy of identities through secure data encryption. Edwards-curve Digital Signature Algorithm (EdDSA) then offers a state-of-the-art mechanism for generating digital signatures, ensuring data integrity and non-repudiation. Finally, the Keccak (SHA-3 precursor) hashing function strengthens the system's security by providing a unique and cryptographically secure message digest for data authentication. An analysis is conducted to explore the system's suitability and effectivity for the proposed application. The findings from this study can contribute significantly to the ongoing efforts of creating a secure digital identity management solution.**

***Keywords— Blockchain, Digital Identity, ECIES, EdDSA, Keccak Hashing Function)***

## I. Introduction

Nowadays, nearly every aspect of our daily lives is digitalized and integrated with many services in order to achieve what we believe as efficient processing. One of such aspect is our own identity. Identity is integral to a functioning society and economy. It is a proper way to identify ourselves and our possessions, eventually enabling us to create thriving, complex civilizations. At its most basic level, identity is a collection of claims about a person, place or thing. For people, this usually consists of first and last name, date of birth, nationality, and possibly ownership claims of material possessions such as house certificate, wallet, driving license, etc. [1].

By migrating and creating another identity to define ourselves in the digital landscape, we are faced with many problems and insecurities such as system leaks, data loss, and especially cyber threats. Even when these problems are being tackled by developments in data security systems, current digital identities are very vulnerable to such problems and existing systems are not well developed yet to serve as a fully robust and secure platform for digital identity protection [2].

However, there is an idea about how digital identity can be managed and protected. A few paper such as [3] and [4] describe the implications of digital identity in a decentralized environment. Both emphasizes the idea of "self-sovereign identity" (SSI), a model in which users own all the rights over their digital identity. In this model, users have control over what happens with their identity, as only they can share it with whoever they want to. In addition to these, the self-sovereign identity model supports the transparency of algorithms and systems, so that everyone can see how they work and have trust in the framework. This concept is very closely linked with blockchain's distributed ledger paradigm.

If a digital identity protection system is developed based on SSI, it is best to have blockchain as its foundation technology. When using blockchain-based technology, SSI must be coupled with robust cryptographic algorithms and hash functions to ensure data security and immutability. Among the best of these algorithms and functions are ECIES, EdDSA, and Keccak function.

This paper aims to discuss a possible solution to digital identity protection using blockchain-based database and secured by using ECIES, EdDSA, and Keccak. By distributing user identities on a blockchain across a decentralized network, the data are made immutable to prevent any unauthorized tampering to every data in the databases. ECIES is used to provide a secure encryption and decryption of user identities, which are then given digital signatures using EdDSA and Keccak. By using this approach, digital identity protection and management provided by the system is expected to ensure the confidentiality, integrity, and authenticity of user identities while upholding non-repudiation and availability of distributed databases.

This analysis contributes to finding a solution to cyber threats against digital identities which also provides a foundation for future usages of ECC-based algorithms and Keccak-like hash functions in cybersecurity.

## II. Literature Review

### A. *Blockchain*

According to [5], a blockchain is a continuously expanding collection of data blocks linked together to form a long chain. This network of connected data blocks represents a distributed ledger that is disseminated over a peer-to-peer network. This distributed ledger contains a collection of digital data that are synced, replicated, distributed, and shared through a peer-to-peer network.

In practice, blockchains will use cryptographic algorithms and a hash function to secure the data inside every block before getting added to the blockchain. Other technologies may include the usage of Merkle-Patricia Trie for a more efficient data integrity verification and changes, consensus algorithms used for determining how people can validate a block before appending, and smart contracts for automatically executing a certain program after conditions in a block is fulfilled. An instance of a blockchain scheme is illustrated in Figure 1.

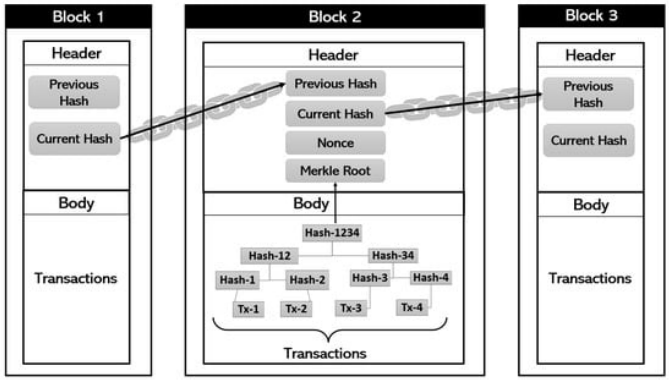


Fig. 1. Blockchain Scheme. [5]

The blockchain is initiated using a Genesis Block which contains the first recorded data. To append a block at the end of the blockchain, the previous block's hash digest is placed inside the block's header along with other important data, such as current block hash digest and nonce. For the sake of simplicity, this implementation will not use consensus algorithms and will simply use pseudo-randomized nonce. The data inside each block will contain a digital identity of different people (refer to Figure 2).

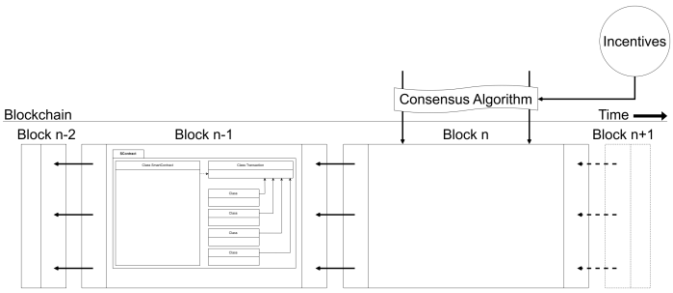


Fig. 2. Appending a Block to Blockchain.

The blockchain will then be distributed among several databases inside a closed decentralized network (refer to Figure 3). This process can be automated and improved by using Raft consensus system [6], however for the sake of simplicity, the distributed databases will be updated at the same time. By doing so, the system's distribution can be disrupted with various attacks, but on a small scale, this method renders no big issue.


Fig. 3. Distributed Databases on a Decentralized Network. [7]

To monitor the state of current blockchain and user identities, we will use a Merkle-Patricia Trie to ensure its integrity and to ease any possible changes to the blockchain. A Merkle Tree is a tree of hash digests repeatedly hashed together to get a root digest which will be placed in every block header. By mixing it with PATRICIA (Practical Algorithm To Retrieve Information Coded In Alphanumeric) principle, we can achieve efficiency in data integrity verification. A trie is a data structure that is used to retrieve a string value by traversing down a branch of nodes that store associated references (keys) that together lead to the end value that can be returned.

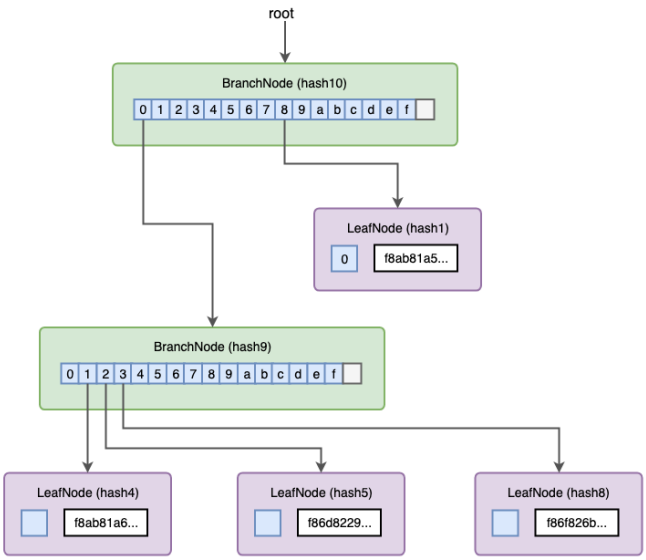


Fig. 4. Merkle-Patricia Trie. [8]

Such implementation of blockchain technology already has a wide range of applications, such as:

- Financial Transactions
  Blockchains can facilitate secure and efficient money transfers through cryptocurrencies, while reducing administration fees and paperwork.
- Smart Contracts
  As mentioned before, blockchains can automate the execution of a program written inside a block on the blockchain. These have many benefits compared to traditional contracts, which include no third-party involvement while also efficient in both time and cost.
- Voting Systems
  The global voting systems can be revolutionized and improved using blockchains due to its decentralized and immutable nature. The usage of blockchain in voting systems will reduce frauds by percentage and increasing voter confidence.
- Digital Identity Management
  As the main reason of this paper, it is clear that by using blockchains, any digital identity protection and management systems can ensure data transparency, security, and giving users more control over their own data.

### B. *Elliptic Curve Cryptography (ECC) based Algorithms*

An elliptic curve is a mathematical construct defined over a field and the points in such curve forms an abelian group. It has several unique characteristics that can be used in cryptography if the field it was defined on is a finite field. There is also ECDLP (elliptic curve discrete logarithm problem) as an intrinsic factor of elliptic curve which all ECC-based algorithms depend on. Let there be points P and Q on the elliptic curve such that $kP = Q$ where $k$ is a scalar. We can define $k$ as the discrete logarithm of Q in base P and when one tries to obtain $k$ by knowing P and Q, it will be significantly harder than knowing $k$ and trying to find one of the points [9].

In elliptic curves, point multiplication is achieved through two methods, namely point addition and point doubling. Point addition (refer to Figure 5) is a method on obtaining a new point on the curve by pulling a line through two input points and determining the intersection of said line with the curve. The point at such intersection is the negative of the desired point. To get the counterpart of a point, said point will be projected symmetrically against the x-axis. If a point is added to its negative counterpart, the result would be the point O, defined as a point at infinity.

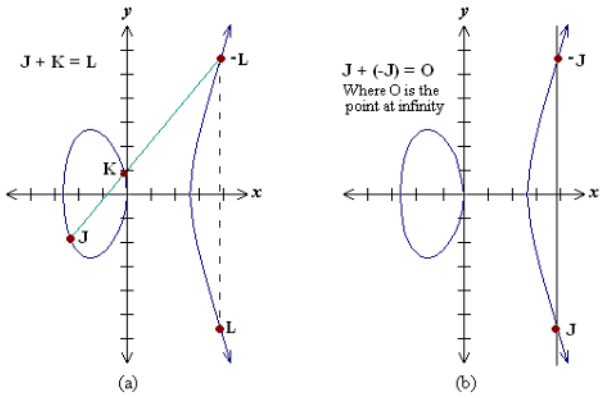


Fig. 5. Point Addition on Elliptic Curves. (a) Point Addition of Two Different Points. (b) Point Addition of a Point and Its Negative Counterpart. [9]

If a point on the elliptic curve is added to itself, it is called point doubling. Point doubling works similarly with point addition, differs only on how the line is drawn. Since there are only one point of interest in the curve, the resulting point is obtained by pulling a line tangent to the curve at that point. The intersection on said line with the curve results in the negative counterpart of the desired point. However, if the input point has y-coordinate of 0, then the resulting point is the point O, located at infinity (refer to Figure 6).

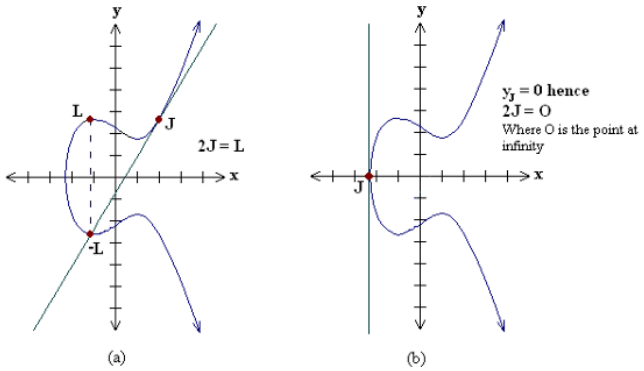


Fig. 6. Point Doubling on Elliptic Curves. (a) Point Doubling if Y-Coordinate is non zero. (b) Point Doubling if Y-Coordinate is zero. [9]

To achieve a cryptographically functional elliptic curve, the curve must be defined over a finite field of a prime number. By using the properties of a finite field, point multiplication on the curve can be done relatively quicker while retaining ECDLP. The resulting algorithms are created specifically for many specialized purposes as shown in Figure 7.

| Application | Protocol |
|---|---|
| Key Negotiation / Key Exchange | Elliptic Curve Diffie-Hellman (ECDH) |
| | Elliptic Curve Menezes-Qu-Vanstone (ECMQV) |
| Encryption / Decryption | Elliptic Curve Integrated Encryption Standard (ECIES) |
| | Elliptic Curve Massey-Omura Encryption (ECMOE) |
| Digital Signing / Signature Verification | Elliptic Curve Digital Signature Algorithm (ECDSA) |
| | Elliptic Curve ElGamal Digital Signature Algorithm (ECEGDSA) |

Fig. 7. Protocols Based on ECC. [10]

ECIES is chosen for its robustness compared to other versions of ECC based cryptosystems. EdDSA (Edwards Curve Digital Signature Algorithm) is better in nature with ECDSA, since it is based on a different type of elliptical curve (refer to Figure 8) and using a different type of signature called the Schnorr's Signature [11]. According to [12], an Edwards curve is a particular form of an elliptic curve that leads to fast, unified, and complete addition formulas. Using points along an inverted or twisted Edwards curve (refer to Figure 9) yields the fastest formulas for adding points on such a curve. This further translates into faster cryptographic operations when using this curve compared to normal elliptic curves.

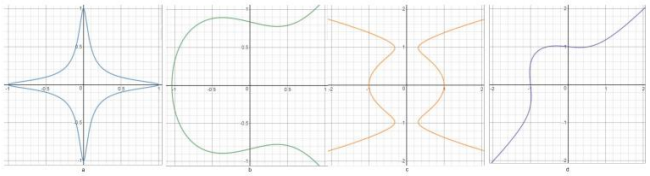

Fig. 8. Forms of Elliptical Curves. (a) Edwards Curve. (b) Weierstrass Curve. (c) Jacobi-quartic Curve. (d) Hessian Curve. [11]

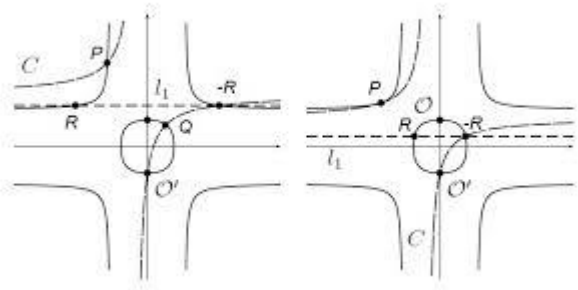


Fig. 9. Point Addition on Twisted Edwards Curve. [13]

By using these algorithms, the blockchain-based database will endure cyberattacks better than using other cryptosystems, such as RSA [14]. It is much faster for larger data sets and provides better security even when using smaller key sizes.

### *C. Keccak Hash Function*

Keccak hash function was crowned as the winner of NIST Hash Function Competition in 2012, making it the basis of a new hash standard called SHA-3. The reasoning behind said competition is that NIST wanted to find a hash function dissimilar from older SHAs, such as SHA-0 and SHA-1 which are proven to be vulnerable to attacks and are deprecated. As SHA-2 are also similar in construction with its precursors (even with no proven successful breaches), NIST was wary about the risks by having no alternative to SHA-2 which has no similar weaknesses compared with previous standards.

Keccak hash function is constructed by using a sponge-like architecture (refer to Figure 10), unlike SHA-2 and its precursors that has similar workflow. However, Keccak also has its own weaknesses, as revealed on tests against SHA-3. The sponge-like construction refers to its function which resembles a sponge by "absorbing" data to the function and then "squeezes" out the results.

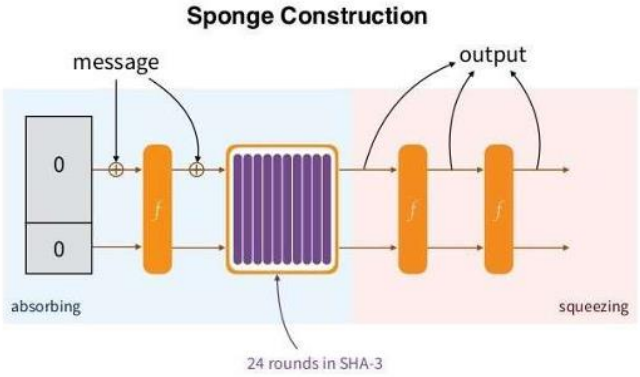


Fig. 10. Sponge Architecture in Keccak Hash Function. [15]

Every time *n*-bit part of the message gets "absorbed" to the function state using XOR operation, it undergoes 24 rounds of processing using the *f* function. The *f* function is comprised of five different components that works sequentially, called θ (Theta), ρ (Rho), π (Pi), χ (Chi), and ι (Iota) respectively. The results of a previous *f* function processing will become the state of the function, accepting new *n*-bit message part to process in the next 24 rounds and the process repeats until there are no more message to "absorb". It is noteworthy that not every part of the state block will accept new message parts. This increases the quality of resulting message digest and preventing collisions among existing digests.

Based on [16], the five subprocesses are explained as the following:

- θ component:
  Consists of a parity computation, a rotation of one position, and a bitwise XOR. This is depicted and formulated in more detail in Equation 1.

$$C[x] = A[x,0] \oplus A[x,1] \ldots \oplus A[x,4] \; for \; 0 \le x \le 4$$
$$D[x] = C[x-1] \oplus ROT(C[x+1],1) \; for \; 0 \le x \le 4$$
$$A[x,y] = A[x,y] \oplus D[x] \; for \; 0 \le x,y \le 4 \quad \ldots\ldots\ldots (1)$$

- ρ component:
  Doing a rotation by an offset that depends on the word position using a different triangular number 0, 1, 3, 6, 10, 15, etc. This is denoted in Equation 2.

$$A[x,y] = ROT(A[x,y], r[x,y]) \; for \; 0 \le x,y \le 4 \ldots\ldots (2)$$

- π component:
  This component is implemented to modify the Keccak columns and lanes position using permutation. This is denoted in Equation 3.

  $$B[y, 2x + 3y] = A[x, y]\ 0 \leq x, y \leq 4 \quad \ldots\ldots\ldots\ldots\ldots\ldots (3)$$

- χ component:
  It consists of five rows of five lanes and implements 25 XOR, 25 AND, and 25 NOT of 64-bit logic gates. This is depicted in Equation 4.

  $$A[x, y] = B[x, y] \oplus ((NOT\ B[x + 1, y])\ AND(B[x + 2, y]))\ for\ 0 \leq x, y \leq 4 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (4)$$

- ι component:
  The final component performs XOR operation between the first lane and the round constant value. This is denoted in Equation 5.

  $$A[0, 0] = A[0, 0] \oplus RC \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (5)$$

After the "absorbing" phase is completed, the function enters the "squeezing" phase in which the result is taken out from the function state. If the digest taken is not yet as long as the desired length, the *f* function will be applied to the state and the current state will be appended to said digest. This may be exhausting computational resources, but it will decrease the chances of a collision even further.

## III. Design and Implementation

When creating a system for digital identity protection, the first step is to implement ECIES, EdDSA, and Keccak Hash before using all of it in the blockchain. Each part of the whole system will be designed and implemented according to current paper needs hence these will be simplified version from what actually being used in real world.

Referring to Figure 11, at first user will access the system through a PC or laptop. User will then decide to generate new keys or to use existing keys. The keys will be used to process the digital identity that user has entered. EdDSA keys will be used to digitally sign the data and includes the signage to block processing. In this process, the block under construction will place the signage with the data, encrypts the data block using ECIES, and then defining its headers (such as previous block digest, merkle root, timestamp, etc.). After going through the consensus algorithm and mined by a miner, the block's last header part is filled (nonce and current block digest). The block is then appended to the blockchain stored inside the device's local database. The database, being distributed among its peer in a decentralized network, will resolve whether its blockchain is the longest and the most agreeable one. If it is, other peers in the network will copy the blockchain and distribute it.

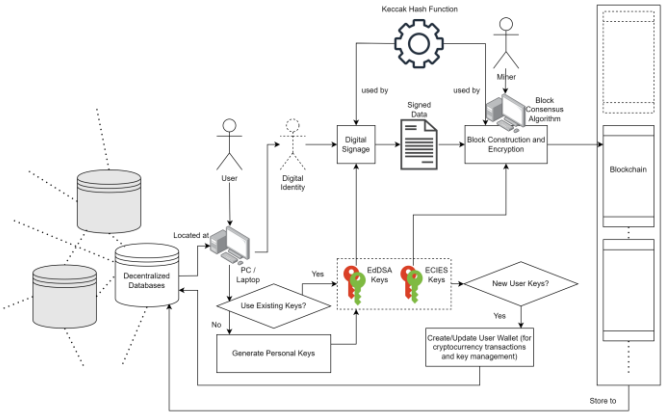


Fig. 11. Digital Identity Protection System (DIPS) Scheme.

For EdDSA, the signing process involves generating two parts of signature, namely sign A and sign B (refer to Figure 12). Creating the signature will require the usage of Keccak Hash Function as well as the Edwards Curve formula and initial parameters. After signing, the data can be verified using the creation of two test proofs that will be compared. If any of the original data is changed or the data owner repudiates, the proofs will not be equal.

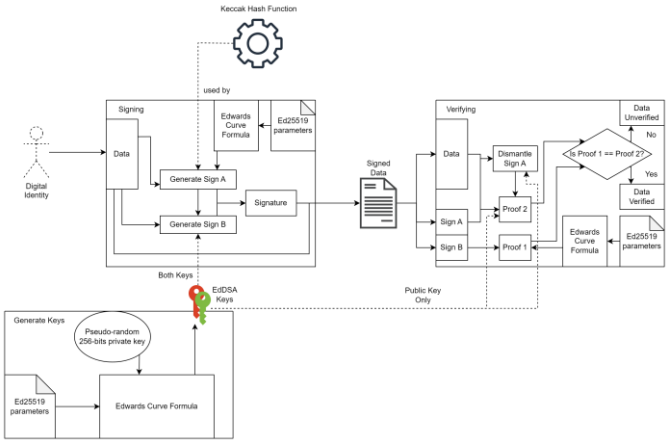


Fig. 12. EdDSA in DIPS Operational Scheme.

By using the scheme in Figure 12, an EdDSA component for DIPS can be implemented using Python language. The following is a mock test of said implementation along with its generated data and processing time.

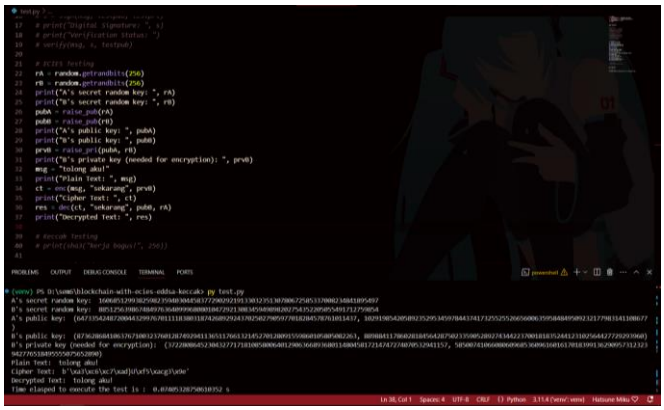

Fig. 13. EdDSA Mock Test.

For ECIES, the encryption process starts after digital signature for the data is created. It involves generating a key pair, namely a random key and a public key as well as a private key for encryption (refer to Figure 14). The key generation will be using normal Elliptic Curve formula and initial parameters. The encryption and decryption process will be using AES on EAX mode and the nonce in the block header. On non-blockchain usage (such as normal message exchange), the keys can be derived as two parts, one being used in actual encryption-decryption process, while the other is used for getting a HMAC to authenticate the message contents.

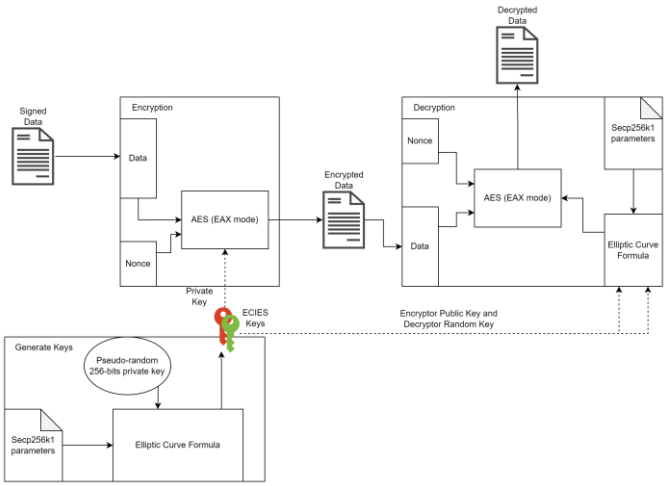


Fig. 14. ECIES in DIPS Operational Scheme.

By using the scheme in Figure 14, an ECIES component for DIPS can be implemented using Python language. The following is a mock test of said implementation along with its generated data and processing time.

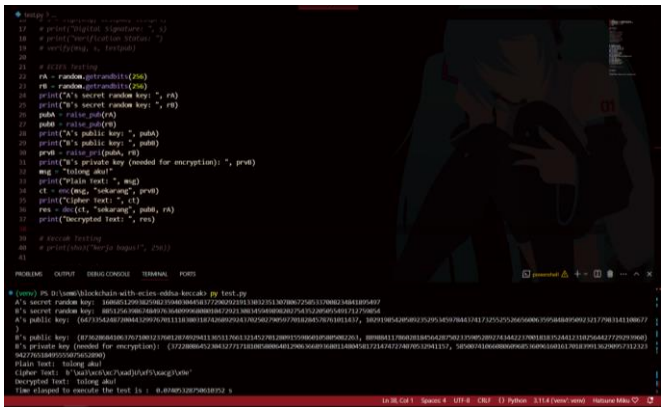

Fig. 15. ECIES Mock Test.

For Keccak Hash Function, the process is exactly as mentioned in Section II. The "absorbing" phase will continue to "absorb" *n*-bit parts of the message until there is none left. The "squeezing" phase will continue to "squeeze" out parts of the digest until it reaches the desired digest length. In between "absorbing" and "squeezing" messages or digests, it will process the current function state (and newly "absorbed" message parts) using an *f* function for 24 rounds. The *f* function will be comprised of five different stages, each jumbling the input data in formulated and calculated ways (refer to Figure 16).

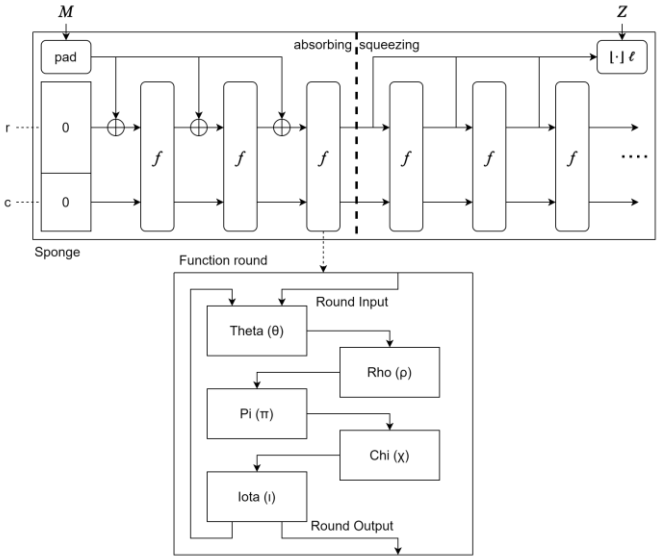


Fig. 16. Keccak in DIPS Operational Scheme.

By using the scheme in Figure 16, a Keccak component for DIPS can be implemented using Python language. The following is a mock test of said implementation along with its generated data and processing time.

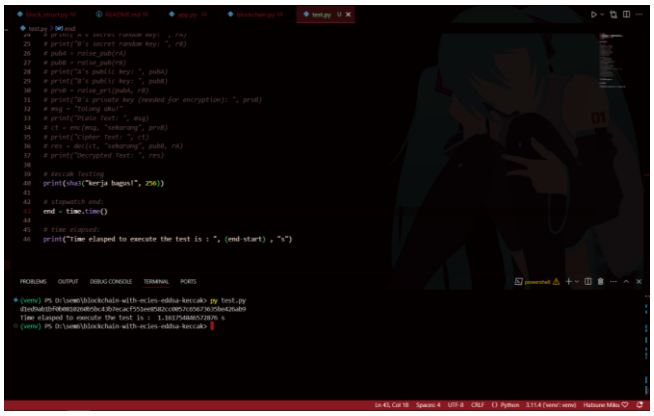

Fig. 17. Keccak Mock Test.

Finally, the blockchain will be generated from a genesis block that contains the first data (usually a dummy data). The block that appears afterwards must be digitally signed, encrypted, and mined by a miner before being appended to the blockchain. This chain will then be distributed among peers on a decentralized network. As mentioned before, it is ideal to use Raft consensus algorithm for database management, however for the sake of simplicity, the blockchain in this paper will only use simple error resolving methods.

## IV. Simulation and Analysis

The simulation starts with user opening the application. User will see the client side of blockchain system in order to simulate a new user creating a wallet (identity) before using the blockchain. There will be a button to generate a new wallet which in turn will generate new pairs of keys for the user (refer to Figure 18).


Fig. 18. DIPS Blockchain Client Side.

If user inputs a public and private key, the system will use that as EdDSA keys (ECIES keys can be generated multiple times). User can proceed to input his/her data into the system and process it to the blockchain. The system will automatically use EdDSA to sign the data and ECIES to encrypt it after the block has been mined. As shown in Figure 19, the user can also play the role of a miner which mines new block that are generated across the network.

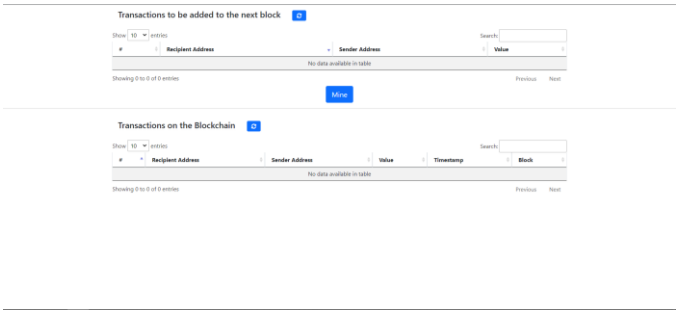


Fig. 19. DIPS Blockchain Miner Side.

The user can now help other users validate their data. This is the part of the system that makes everyone can see who sent the data but not the data itself. However, anyone can help to secure the data by using their own private key and the owner's public key in order to create a safe and trusting network of identities. A user may change his/her data, but it will need a consent from the miner that validated the block.

While working with this system, a timer is prepared to record the time elapsed for data inputs of size 10, 100, and 1000 (through console for looping). The data shows that there are little to no significant increase in time and cost taken to process data on an even larger scale. The following is the data gathered from tests and categorized based on the components used in the system. In order to get a greater grasp of each component, Keccak will not be used in EdDSA and replaced by Python built-in hash library.

TABLE I
Test Results

| Subsystems | Time Elapsed for Each Data Sizes | | |
|---|---|---|---|
| | 10 | 100 | 1000 |
| EdDSA | 0.261 s | 1.724 s | 4.632 s |
| ECIES | 0.075 s | 1.829 s | 3.908 s |
| Keccak | 1.103 s | 15.074 s | 91.031 s |

From the results, it can be determined that Keccak is significantly slower than other subsystems. This can be answered by the implementation of full repeated 24 rounds of Keccak function, whereas SHA-256 used from Python hash library on ECIES and EdDSA takes far less computation. However, the security levels might be compromised in the future if SHA-256 is used for this system. Overall, it can be ruled out that ECIES and EdDSA provides a great safety level at small time and cost even on large scale data processing.

Meanwhile, Keccak Hash Function might not be the best solution for the system as it has quite large processing time when compared with other hash standards. Keccak might provide an extra layer of security, but the time and cost it needs exceeded the acceptable threshold as mentioned by Devi [3].

Other test should be conducted in the future in order to get a more comprehensive view about the usage of each subcomponent. There could also be other solutions to the same problem that uses different technologies and can be compared with one another in order to determine the best possible solution to solve digital identity management and protection.

## V. Conclusion

Digital identity protection is paramount in today's cybersecurity landscape. A system comprised of blockchain technology, ECIES, EdDSA, and Keccak could safeguard digital identities. While the combination offers promising security features, a comprehensive analysis is crucial to assess its real-world applicability.

The analysis conducted in this paper is limited. Nevertheless, it can give a general idea about which underlying technologies are fitting for a blockchain based solution to this problem. The blockchain technology itself does not being tested as the scope is being limited to a small peer-to-peer network of databases. Mathematically, EdDSA and ECIES are both superior when compared to other algorithms such as ECDSA and EC-ElGamal respectively.

The test shows that EdDSA and ECIES are proven to be fast, robust, and capable of handling large amounts of data. Meanwhile, Keccak Hash Function is theoretically better and more resistant to attacks than other SHAs. This does not prove to be true in the test, as the Keccak Hash used in the system uses large amount of computational time and resources on a level that overthrows its better security level.

Although an extra layer of protection is very much needed in the problem focused, a slow hashing function will slow down a significant percentage of this DIPS system. Therefore, Keccak Hash Function can be considered not ready or replaceable with other SHA functions that are quicker and easier to implement.

## Source Code

Here is the link to a GitHub repository about the implementation of digital identity protection using blockchain-based database, ECIES, EdDSA, and Keccak Hash:

https://github.com/toper664/blockchain-with-ecies-eddsa-keccak

## Video Link At Youtube

Here is the link to a YouTube video about the analysis and implementation of Digital Identity Protection:

https://youtu.be/-qQKK5Iwauu

## References


[1] L. J. Camp, "Digital identity," *IEEE Technology and Society Magazine*, vol. 23, no. 3, pp. 34–41, Feb. 2004, doi: 10.1109/MTAS.2004.1337889.

[2] "Digital Identity On the Threshold of a Digital Identity Revolution," 2018.

[3] S. Devi, S. Kotian, M. Kumavat, and D. Patel, "Digital Identity Management System Using Blockchain." [Online]. Available: https://ssrn.com/abstract=4127356

[4] A. Giannopoulou, "Digital Identity Infrastructures: a Critical Approach of Self-Sovereign Identity," *Digital Society*, vol. 2, no. 2, p. 18, Aug. 2023, doi: 10.1007/s44206-023-00049-z.

[5] M. Krichen, M. Ammi, A. Mihoub, and M. Almutiq, "Blockchain for Modern Applications: A Survey," *Sensors*, vol. 22, no. 14, p. 5274, Jul. 2022, doi: 10.3390/s22145274.

[6] D. Ongaro and J. Ousterhout, "In Search of an Understandable Consensus Algorithm (Extended Version)."

[7] A. C. Careja and N. Tapus, "Digital Identity Using Blockchain Technology," in *Procedia Computer Science*, Elsevier B.V., 2023, pp. 1074–1082. doi: 10.1016/j.procs.2023.08.090.

[8] L. Zhang, "Ethereum Merkle Patricia Trie Explained."

[9] A. Ms and A. C. In, "Elliptic Curve Cryptography-An Implementation Tutorial Elliptic Curve Cryptography An Implementation Guide."

[10] H. Alrimeih, "Fast and Flexible Hardware Support for Elliptic Curve Cryptography over Multiple Standard Prime Finite Fields," 2012.

[11] J. Guruprakash and S. Koppu, "An Empirical Study to Demonstrate that EdDSA can be used as a Performance Improvement Alternative to ECDSA in Blockchain and IoT," *Informatica (Slovenia)*, vol. 46, no. 2, pp. 277–290, Jun. 2022, doi: 10.31449/inf.v46i2.3807.

[12] F. Morain, "Edwards curves and CM curves," Apr. 2009, [Online]. Available: http://arxiv.org/abs/0904.2243

[13] M. Ashraf and B. Kırlar, "INTERNATIONAL JOURNAL OF INFORMATION SECURITY SCIENCE On the Alternate Models of Elliptic Curves."

[14] J. Bao, "Research on the Security of Elliptic Curve Cryptography," 2022.

[15] S. Sharma, . L., and S. Khanum, "Performance analysis of SHA 2 and SHA 3," *SSRN Electronic Journal*, 2022, doi: 10.2139/ssrn.4068861.

[16] H. Mestiri, I. Barraj, and M. Machhout, "A High-Speed KECCAK Architecture Resistant to Fault Attacks," in *2020 32nd International Conference on Microelectronics (ICM)*, IEEE, Dec. 2020, pp. 1–4. doi: 10.1109/ICM50269.2020.9331792.


## PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 12 Juni 2024

Christopher Febrian Nugraha / 18221115