Privacy-preserving credential smart contracts using Zokrates

Geunyoung Kim, Yunsik Ham and Jaecheol Ryou*
Department of Computer Science, Chungnam National University
Daejeon, South Korea
[e-mail: gykim@cnu.ac.kr]
*Corresponding author: Jaecheol Ryou

Received March 31, 2024; accepted June 3, 2024;
published August 31, 2024

Abstract

The need for secure user authentication in blockchain-based applications has been growing with the increased adoption of Decentralized Identity (DID) credentials in blockchain. Zokrates, a tool designed to protect user privacy within smart contracts, had a limitation in that it could not accept authenticated user information such as credentials, only allowing the use of manually inputted data. In this paper, we propose a smart contract system that securely validates DID credentials to overcome the limitations of traditional centralized authentication systems. This system ensures the safe identification of users within blockchain-based applications by authenticating their identities in a trusted manner within the blockchain. As the demand for user authentication in blockchain rises, this paper emphasizes the significance of a blockchain-based identity verification system that guarantees both privacy and security. Leveraging the Zero-Knowledge Proof method and utilizing the Zokrates tool, this innovative approach aims to provide solutions for the digital identity verification process, thereby expanding the scope of blockchain technology applications. Moreover, we also provide a CLI for each entity. We help anyone who wants to authenticate their identity using the tool to safely verify it on-chain.

Keywords: Blockchain, Privacy-preserving, Credential, Zero-knowledge proof

A preliminary version of this paper was presented at ICONI 2023, and was selected as an outstanding paper. This version includes a detailed analysis of zero knowledge proof generation and verification contracts in Zokrates. This work was supported by the Ministry of Education of the Republic of Korea and the National Research Foundation of Korea.

http://doi.org/10.3837/tiis.2024.08.020 **ISSN : 1976-7277**

1. Introduction

The emergence of Bitcoin[1] in 2009 marked the beginning of the blockchain technology era, followed by Ethereum[2] in 2014, which significantly expanded the capabilities of blockchain. Among these developments, Non-Fungible Tokens (NFTs) have emerged as particularly important. The concept of NFTs was first introduced in early 2017 through projects like CryptoPunks, a variant of ERC-20[3], and CryptoKitties, which utilized ERC-721[4], gaining such popularity that they overwhelmed the Ethereum network. Meanwhile, virtual asset exchanges worldwide are increasingly adopting Know Your Customer (KYC) procedures as part of their efforts to combat money laundering and terrorist financing, both domestically and internationally. KYC is a mandatory procedure under regulations governing the reporting and use of specific financial transaction information. Consequently, the Financial Action Task Force (FATF) recently emphasized the importance of verifying users' identities not only in virtual asset exchanges but also in Web3 fields such as Decentralized Finance (DeFi) and NFTs due to their decentralized nature[5]. However, within the blockchain ecosystem, verifying identities poses a challenge due to the anonymity of wallet addresses. Although Decentralized Identity[6] (DID) have gained traction for user authentication, verifying DIDs still requires off-chain methods. Therefore, there is a need for on-chain authentication solutions to address this issue. However, given the transparent nature of the blockchain, there is a risk of exposing users' identities. To mitigate this, tools like Zokrates[7] offer Selective Disclosure, allowing users to reveal only the information they choose. However, a drawback is that Zokrates currently does not support DID processing. In this paper, we propose an on-chain authentication method for DIDs, ensuring user privacy and enabling various extensions in Web3. We provide CLI utilizing Zokrates, which offers separate functionality for each entity, enhancing usability. Additionally, we prioritize compatibility with W3C standards by utilizing credentials specified by W3C instead of separate credentials.standards by utilizing Credentials specified by W3C, rather than relying on separate credentials.

2. Background

In the blockchain ecosystem, there have been various cases of user authentication. However, most of them involve off-chain authentication of users before providing services to authenticated users on-chain, rather than conducting authentication entirely on-chain. This setup makes it challenging to claim transparent user authentication. Additionally, while there are tools supporting zero-knowledge proofs within smart contracts, they have been limited by the inability to perform signature verification.

2.1 DID & Credential

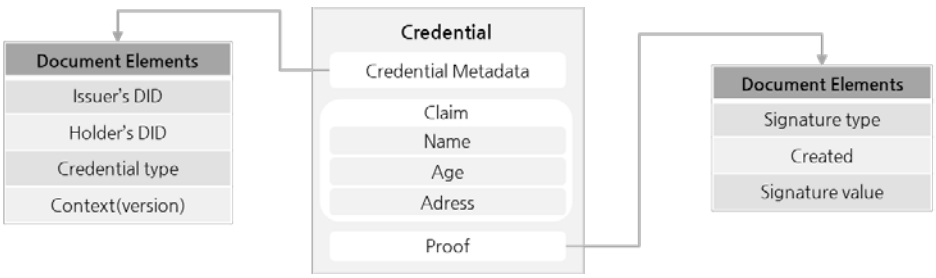

Fig. 1. Components of a credential

DID is a system designed to manage identity information within a decentralized framework, eliminating the necessity for centralized registration mechanisms. Grounded in distributed ledger technology, DID enables individuals to independently manage proof of identity, determining the scope and recipients of information submissions. It functions as a decentralized digital identity verification system, highlighting individual control over personal information, ensuring data sovereignty in an era where safeguarding personal data is crucial in the data economy. The versatility of DID technology is apparent in its applications, particularly in scenarios requiring adult verification. Credentials can be created through DID as depicted in **Fig. 1**. Credentials can typically be matched with a resident registration card. Typically, presenting a resident registration card reveals extensive personal details such as birthdate, address, and registration number. DID addresses this privacy concern by allowing the proof of being an "adult aged 19 or above" without disclosing the exact age. While DID is applicable in offline settings, challenges arise when implementing it online, particularly in the Web3 environment. Examining at projects that have attempted identity authentication within the blockchain ecosystem using DID, Sovrin[8] implemented a decentralized PKI using DID, but lacks measures to protect user privacy on-chain. [9],[10] propose methods using decentralized DID for smart contracts, but they lack clarity on on-chain signature verification and credential management. [11] suggests the issuance and verification process of credentials, but the drawback lies in the need to verify DID for each service. In [12], the paper presents an identity authentication system utilizing Non-transferable NFT and DID, but it does not mention the usage of Zokrates, nor does it display experimental results for each method.

2.2. Serto

In 2021, ConsenSys-operated uport, recognized for its contributions to widely-used blockchain software like MetaMask and Truffle, underwent a separation, leading to the introduction of Serto[13], a low-code, decentralized identity solution, facilitates the complimentary creation and issuance of DID and VC. Notably, it has recently unveiled a protocol for place through this search engine, it operates in an off-chain manner. Information related to DIDs is stored on the Serto server, requiring a connection with a centralized server for information verification. This integration ensures a robust system for proving NFT ownership and managing decentralized identities, aligning with the broader trends in blockchain technology in **Fig. 2**.

Fig. 2. Serto's DID process

2.3 Tbdex

In the latter half of 2021, Square, the American financial services and mobile payments

company headquartered in San Francisco, California, introduced a white paper titled tbdex[14]. Operating within the realm of DeFi as an application of Web3, tbDEX serves as a decentralized protocol facilitating the exchange between fiat currency and cryptocurrency. The protocol employs DID for user identity verification before executing transactions.Within this framework, there exists a concept called PFI(Participating Financial Institution), representing entities engaged in the exchange between fiat currency and cryptocurrency. Fintech companies, banks, or other financial institutions can seamlessly participate as PFIs by managing blockchain nodes without undergoing a specialized approval process. As depicted in **Fig. 3**, the user communicates transaction details to the PFI, and once transaction terms are negotiated, the user transmits the DID Credential. The PFI then verifies the credential, facilitating the completion of the transaction. While the specific DID format remains undisclosed, the approach involves utilizing credentials for user identity verification. This also has the drawback of relying on centralized banks to verify users' DIDs off-chain.

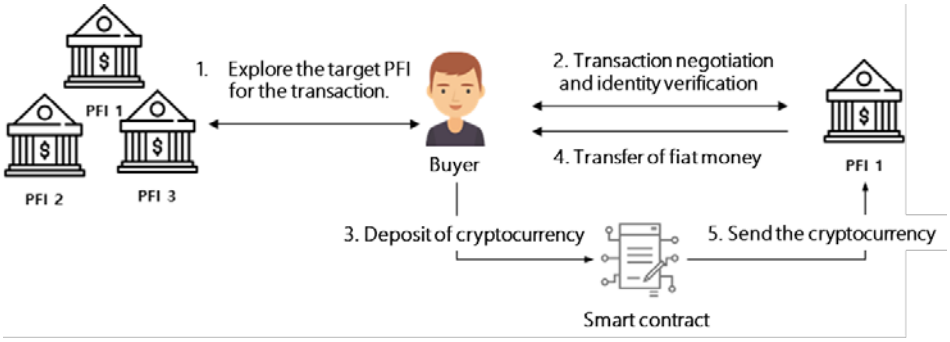

Fig. 3. Tbdex's process

2.4 ZKP

In order to maintain the decentralized nature of blockchain while providing anonymity and privacy protection, zero-knowledge proof (ZKP) can be utilized. ZKP enable verification of a fact without revealing the secret value, thereby offering privacy protection. For example, the cryptocurrency Zcash[15] provides anonymity by proving the validity of transactions without disclosing transaction information such as sender, receiver, and balance. In addition to cryptocurrencies like Zcash, there is a growing body of research applying zero-knowledge proofs in blockchain to enhance privacy protection. A typical ZKP framework consists of three distinct phases[16]:

- ✓ Witness Phase: Initially, the prover formulates a proof based on the statement and sends it to the verifier.
- ✓ Challenge Phase: Subsequently, the verifier generates a challenge based on the proof received and forwards it to the prover.
- ✓ Response Phase: Finally, the prover addresses the challenge and sends the response back to the verifier.

ZKP is a foundational tool in contemporary cryptography, garnering significant attention for its capacity to tackle crucial security and privacy challenges. Its utility spans across diverse domains, encompassing digital identity [17], blind signatures [18], and attribute verification [19].

2.5 Zokrates

In 2016, TU Berlin introduced Zokrates, a tool designed to facilitate ZKP within the Ethereum ecosystem. Illustrated in **Fig. 4**, this tool empowers users to generate proofs using ZKP techniques and automatically produces verifiable contracts for proof validation. This enables the verification of proofs on-chain through smart contracts, allowing anyone to authenticate a user's proof. Zokrates utilizes the zk-snark[20], ZKP algorithm. Due to the characteristics of zk-snarks, generating proofs incurs significant costs, while verification operates in constant time, O(1). This makes it a suitable algorithm for blockchain environments. Proofs are generated off-chain by entities external to the blockchain, while verification occurs on-chain within the blockchain. This ensures low costs within the on-chain environment, making it appropriate for use in smart contracts. Consequently, Zokrates proves to be a valuable tool for on-chain KYC support. However, a drawback exists as the tool does not accommodate DID Credential due to its utilization of a proprietary language for proof creation.

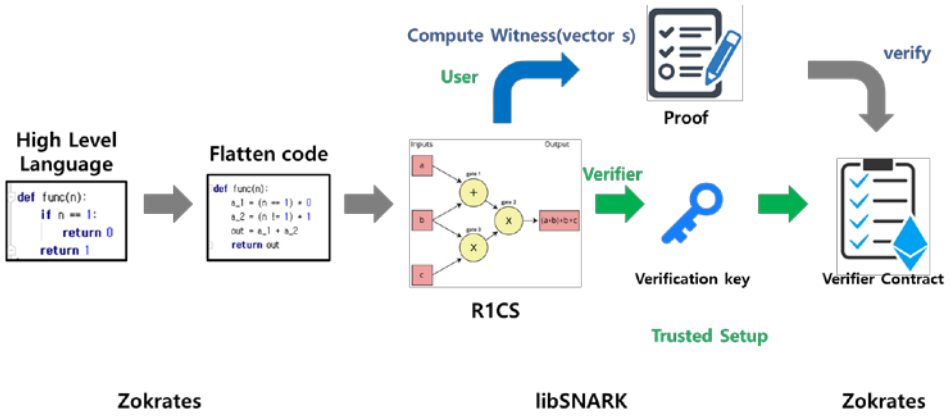

Fig. 4. Zokrates process

3. Privacy-preserving Credential Model

Using Zokrates, the on-chain Privacy-preserving Model is depicted in **Fig. 5**.

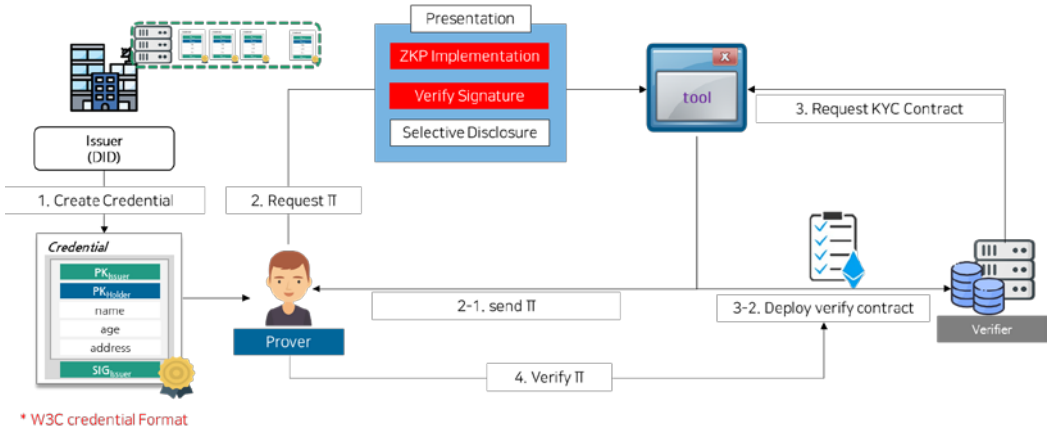

Fig. 5. Privacy-preserving model

1. Prover receives a credential generated by the DID Issuer
2. Prover generates Proof(π) based on Credential using zokrates tools.
3. Create a smart contract capable of verifying the Proof(π).
4. Verify the Proof(π) created through the smart contract.
5. Simultaneously with the verification of Proof(π), the user is verified by using the service.

3.1 Program Structure

The configuration of a tool utilizing Zokrates for generating user proofs and providing Smart Contracts is as depicted in **Fig. 6**.

- ✓ zokrates_credential_cli: A command-line interface that enables users to easily interact with the core logic for issuing, proving, and verifying credentials.
- ✓ zokrates_credential_core: Contains the Rust modules that implement the core functionality of the credential system.
- ✓ zokrates_crypto: migrating the zokrates_pycrypto Library used for generating and verifying signatures.

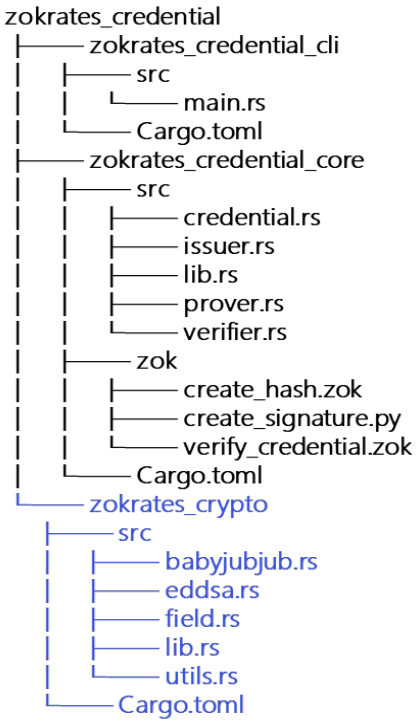

Fig. 6. Structure of Program

The utilization of Zokrates_CLI for each entity is depicted in **Fig. 7**. The Issuer, Prover, and Verifier each utilize Zokrates to create signatures, generate Smart Contracts for verification, and produce proofs.

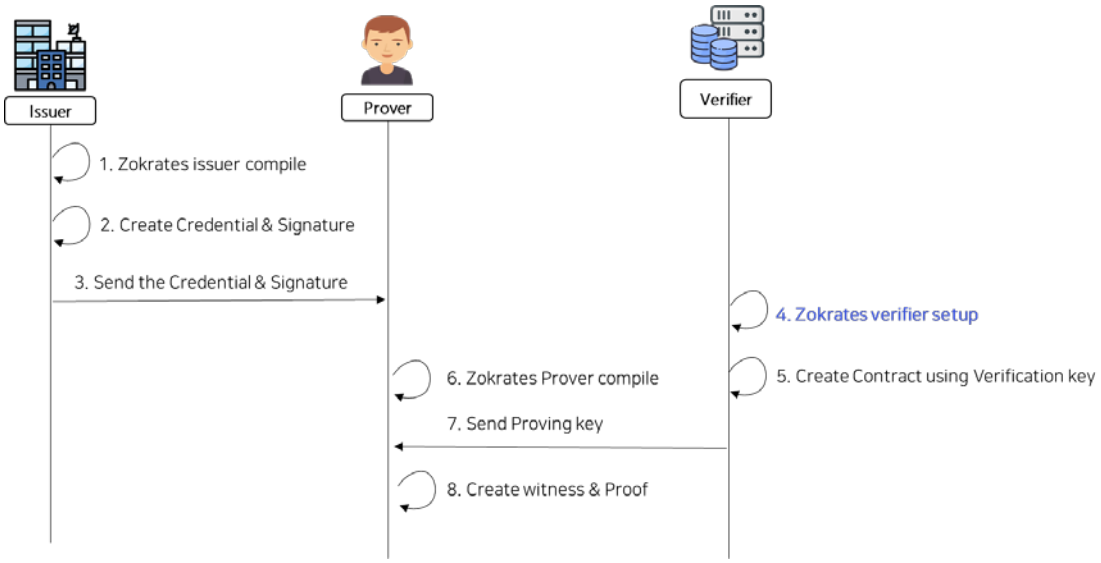

Fig. 7. Process of each entity

3.2 Issuer

The Issuer is responsible for creating credentials for users. As the Issuer ensures the user's identity, the public key used for credential signing is publicly available, and the Issuer's list comprises entities trusted by the Whitepaper or anyone deemed trustworthy. Initially, the Issuer receives information from the Prover and utilizes the Credential format provided by W3C to create the credential. The JSON format of the created credential is depicted in **Fig. 8**.


```
{
  "@context": "88241951271921521112181751841041971331",
  "age": 21,
  "alumni_of": "16121127659931491902152421975228180147",
  "credential_subject": "3155104208121168164210255877110514419 4",
  "exp": "82191332011610815121245701852610020218",
  "id": "153177203205261451371631392139169523 20",
  "issuance_date": "11510212741981143024020720257200291957",
  "issuer": "18917880152136171219259124969219219623",
  "name": "11114352146221731961242531144812432231",
  "student_number": "111143521462217319612425311448124322 31",
  "type": "76982491382252091481301881465616122235"
}
```


Fig. 8. Process of each entity

To prove that the credential was created by the Issuer, the Issuer's signature is included. To generate the signature, the function create_signature is executed based on the credential values. Here, the eddsa algorithm utilizing the BN128 curve[21], is employed. As a result, the Signature and public key are outputted. These values are then passed to the Prover, and later utilized by the Prover when generating the proof off-chain using Zokrates.

3.3 Verifier

The Verifier is responsible for issuing smart contracts to verify the user's identity. When executing the verifier setup through Zokrates, both the proving key and verification key are generated. The proving key is passed to the Prover to create proofs, while the verification key is used to generate a smart contract for verifying the information the Verifier intends to validate based on the user's credential. In this paper, to verify the user's adult status, the Verifier conducts both signature verification of the credential and checks whether the user's actual age matches the officially provided age received from the Issuer. This addresses the issue in traditional Zokrates where users could generate proofs using false inputs. By requiring users to input only the age provided in the credential received from the Issuer, their identity is ensured without any illicit actions. Although the generated smart contract is publicly accessible, only proofs satisfying specific conditions can be verified.

3.4 Prover

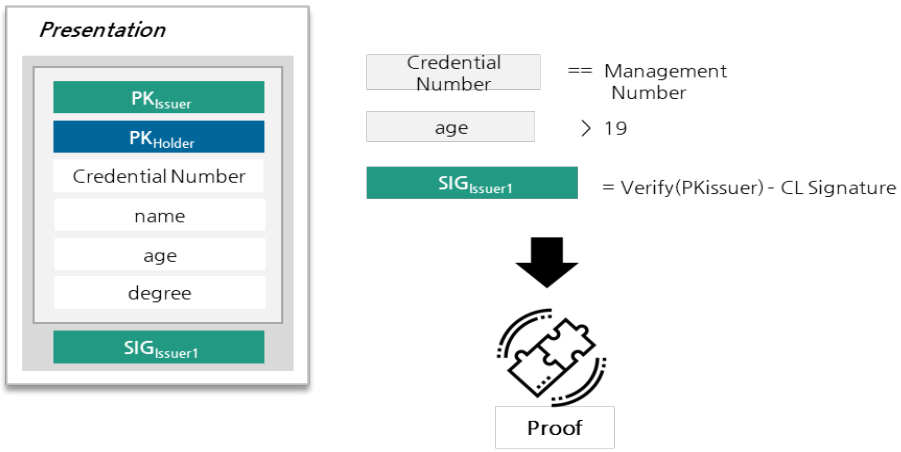

Fig. 9. Process of each entity

The Prover receives verification through the Smart Contract generated by the Verifier for the information they intend to prove. Instead of disclosing their information directly, they provide a proof containing information similar to that depicted in **Fig. 9**. To generate the proof, the Prover needs to receive the proving key from the Verifier. Before creating the proof, the Prover compiles the necessary arguments for the Smart Contract and generates a witness by inputting credential values. Subsequently, the Prover creates the proof based on the witness. The generated proof is depicted as shown in **Fig. 10**.

```
{
  "scheme": "g16",
  "curve": "bn128",
  "proof": {
    "a": [
      "0x0acf41a030e02a129bdfaaa49def09124b964e253aaf563781be12c0a962d9dd",
      "0x157eaf5510e9a8c6eb9e7a6fdabf5203a3171479d3cb1f74a1e974374d8f2084"
    ],
    "b": [
      [
        "0x2d46432de72e3c7ae886d743c925ac3fb9b2bf3a98ef55f7834d9492da898675",
        "0x23954562a22433dad3eed6a2bf2bcd99e2a4786fa3028da6749426c7b027b3af"
      ],
      [
        "0x02cfa3b926600a9243ed980099cb634e7e6594ca468e413d761fd522be50436a",
        "0x0df9b0a5c3491d9afe4dc773c957b2f10ce3c99be16c6c504313dcf80d657fd5"
      ]
    ],
    "c": [
      "0x23ae53ecc0dbc6796566788675f5c16f75167947c10f27906e6c4878178aded5",
      "0x00519f0f6d013b7701388c4a41244561b00447fa2ec93fc009a7f8c4ac491e1e"
    ]
  },
  "inputs": []
}
```


Fig. 10. Process of each entity

3.5 Usage of the Zokrates tool

To generate proofs and contracts, we require a tool that takes credentials as input and produces proofs and contracts. The Issuer, to create a signature using the credentials, hashes the elements of the credential and utilizes the message to generate a signature. Later, for verification, the Issuer utilizes the verifyEddsa function, as shown in **Table 1**, to verify the generated value using the received signature and credential hash values. If any intermediate credential value changes during this process, the value of signature_valid will not be true, preventing the manipulation of false credentials or the Issuer's signature. The Zokrates code for signature verification and age verification, taking credentials as input, is as depicted in **Table 1**.

Table 1. Zokrates code for verification contracts.

```
def main(private field context_hash , private field[2] R, private field S) {

    // To verify changes based on the credential value, hashing operations are performed.

    field[2] first_hash = sha256packed([context_hash, age, alumni_of_hash, credential_subject_hash]);

    field[2] second_hash = sha256packed([exp_hash, id_hash, issuance_date_hash, issuer_hash]);

    field[2] third_hash = sha256packed([name_hash, student_number_hash, type_hash, first_hash[0]]);

    field[2] final_hash = sha256packed([second_hash[1], third_hash[1], third_hash[0], first_hash[1]]);

    // To sign the hashed value, store it in Message format.

    u32[8] M0 = unpack256u(final_hash[0]);

    u32[8] M1 = unpack256u(final_hash[1]);

    // Signature: (R, S), Public Key: A, Memo: M0, M1

    field[2] A = [14897476871502190904409029696666322856887678969656209656241038339251270171395,
                  16668832459046858928951622951481252834155254151733002984053501254009901876174];

    // Perform signature verification and age verification

    bool signature_valid = verifyEddsa(R, S, A, M0, M1, BABYJUBJUB_PARAMS);

    assert(signature_valid == true);

    assert(age >= 20);

    return;

}
```

4. Performance

In the M2 Mac OS Sonoma 14.3.1, Ram 24GB environment, experiments were conducted assuming the presence of Issuers, Provers, and Verifiers. The time taken to compile using the Groth16 scheme from the zk-snark algorithm supported by Zokrates, as well as the setup time to generate proving keys and verification keys, were measured, as depicted in **Fig. 11**. This process was conducted by the Verifier and occurred off-chain, hence it does not affect blockchain performance. The performance was compared as the number of signatures and range proofs increased from 1 to 2, 3, and 5.

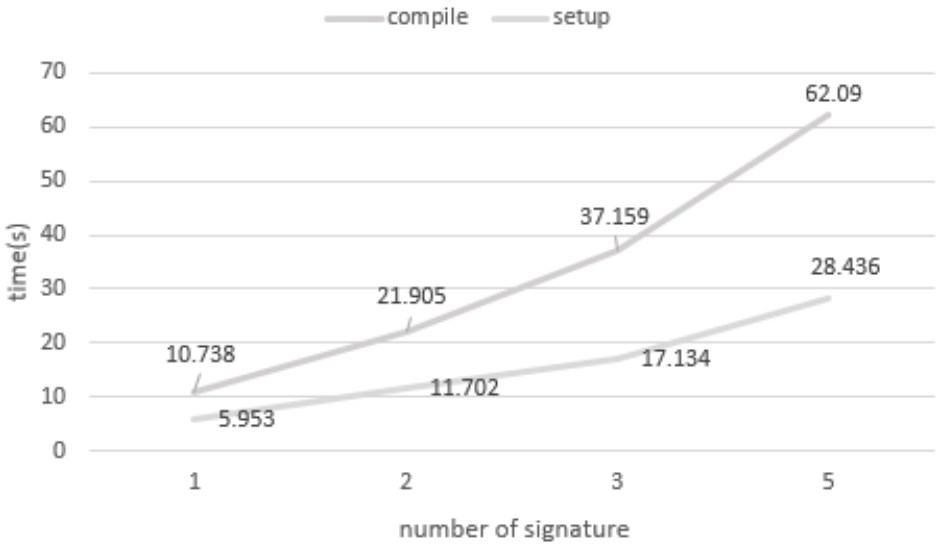

Fig. 11. Compile & Setup performance

Furthermore, performance measurements were conducted for the process where Provers utilize the hash value of the credential and the issuer's public key to generate witnesses and proofs as depicted in **Fig. 12**. Since this process occurs off-chain, it does not affect blockchain performance. It was observed that as the number of signatures increased, the generation time of proofs increased linearly.

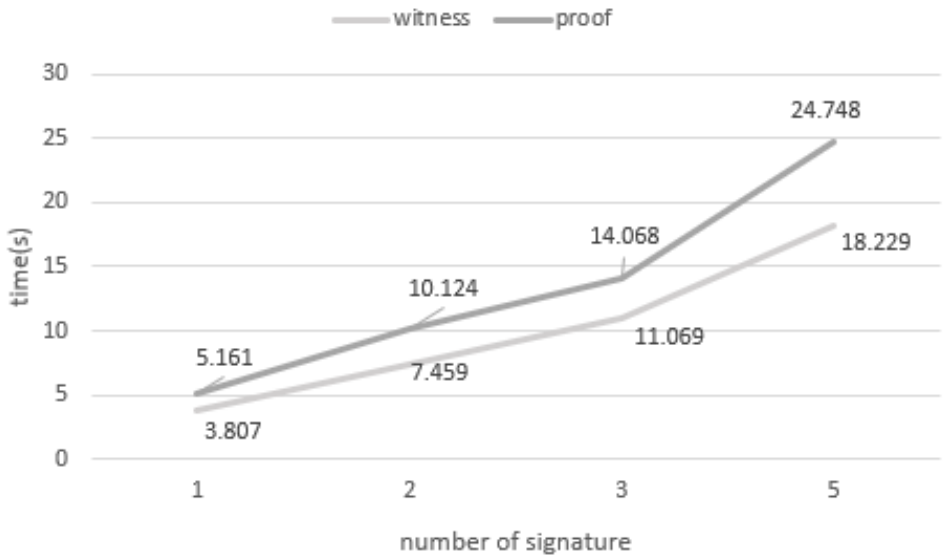

Fig. 12. Witness & Proof performance

In contrast, when verifying whether a user has provided the correct proof through a smart contract function, it was observed that regardless of whether 1, 2, 3, or 5 proofs were verified, 208967 gas was consumed each time, as depicted in **Fig. 13**. Since the computational model used in ZK-SNARKs utilizes R1CS, we can confirm that the verification time is always constant, specifically O(1).

Fig. 13. Verification contract fee

We are able to demonstrate adult authentication through signature verification and range proof using Zokrates. For both Issuers and Provers, while generating proofs off-chain to prove identity, we observed O(nlong) growth in computational complexity. However, it was proven that verifying proofs generated within smart contracts takes constant time. This development allowed us to create a tool that can verify users' credentials on-chain at a low cost while simultaneously protecting their privacy. As a result, we obtained meaningful outcomes.

5. Conclusion

The advantage of our proposed tool is its ability to generate zero-knowledge proofs from DID credentials, a capability Zokrates lacks. Additionally, this tool features a ZKP library specialized for ZK-SNARK, parsing user credentials to create proofs and contracts. Moreover, it offers the advantage of providing a program CLI for users, offering an environment tailored to each entity. By leveraging this tool, user authentication can be conducted on-chain, allowing for various applications in DApps. For instance, by issuing NFTs or SBT (Soul Bound Token) to Provers who submit correct proofs, user authentication can be marked as complete, enabling services only to those who possess the corresponding tokens, even in the Metaverse or DeFi. However, it's important to note that this tool does not represent a fully decentralized system. In this framework, user identity authentication relies on utilizing DID credentials, necessitating support from trusted issuers when generating DID credentials. Furthermore, the current smart contract operates solely on the Ethereum Virtual Machine (EVM), limiting support to the Ethereum blockchain and lacking compatibility with various blockchain platforms. Therefore, research is needed to support diverse blockchain ecosystems in the future, along with exploration into fully decentralized systems for identity authentication on-chain, not reliant on DID.

Acknowledgement

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (RS-2023-00229400, Development of user authentication and privacy preserving technology for a secure metaverse environment) and by research fund of Chungnam National University.

Reference

[1] S. Nakamoto, Bitcoin: A Peer-to-Peer Electronic Cash System, 2008. Article(CrossRefLink)
[2] V. Buterin et al., "Ethereum whitepaper," GitHub repository, 1, 22-23, 2014. [Online]. Available: https://static.peng37.com/ethereum_whitepaper_laptop_3.pdf
[3] ERC-20: Token Standard.
[Online] Available: https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md
[4] ERC-721: Non-Fungible Token Standard.
[Online] Available: https://github.com/ethereum/ EIPs/blob/master/EIPS/eip-721.md
[5] The Financial Action Task Force (FATF), Virtual assets and virtual asset service providers, 2021. [Online]. Available: https://www.fatf-gafi.org/content/dam/fatf-gafi/guidance/Updated-Guidance-VA-VASP.pdf.coredownload.inline.pdf
[6] REED, Drummond, et al. Decentralized identifiers (dids) v1. 0. Draft Community Group Report, 2020. [Online]. Available: https://www.w3.org/TR/did-core/
[7] J. Eberhardt, S. Tal, "ZoKrates - Scalable Privacy-Preserving Off-Chain Computations," in *Proc. of 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pp.1084-1091, 2018. Article (CrossRef Link)
[8] D. Khovratovich, J. Law, "Sovrin: digital identities in the blockchain era," Github Commit Jasonalaw, Oct. 2017. Article (CrossRef Link)
[Online]. Available: https://sovrin.org/wp-content/uploads/AnonCred-RWC.pdf
[9] M. Ramachandran et al., "Towards Complete Decentralised Verification of Data with Confidentiality: Different ways to connect Solid Pods and Blockchain," in *Proc. of WWW '20: Companion Proceedings of the Web Conference 2020*, pp.645-649, 2020. Article (CrossRef Link)
[10] M. Casonato, Owning your data through Self-Sovereign Identity: agents implementation for Verifiable Credentials interaction, 2021.
[Online]. Available: https://thesis.unipd.it/handle/20.500.12608/34924
[11] R. Mukta et al., "Blockchain-Based Verifiable Credential Sharing with Selective Disclosure," in *Proc. of 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pp.959-966, 2020. Article (CrossRefLink)
[12] G. KIM, J. Ryou, "Digital Authentication System in Avatar Using DID and SBT," *Mathematics*, vol.11, no.20, 2023. Article (CrossRefLink)
[13] Uport-Project, Jul. 2021. [online] Available: https://github.com/uport-project/specs.
[14] Jack Doresy, tbDEX: A Liquidity Protocol v0.1.
[online] Available: https://www.coursehero.com/file/146641400/whitepaperpdf/
[15] D. Hopwood et al., Zcash Protocol Specification, GitHub: San Francisco, CA, USA, 2016, 4.220: 32. [Online]. Available: https://raw.githubusercontent.com/zcash/zips/master/protocol/protocol.pdf
[16] M. Abdalla et al., "From Identification to Signatures via the Fiat-Shamir Transform: Minimizing Assumptions for Security and Forward-Security," in *Proc. of Advances in Cryptology – EUROCRYPT 2002: International Conference on the Theory and Applications of Cryptographic Techniques, 2002 Proceedings*, pp.418-433, 2002. Article (CrossRefLink)
[17] X. Yang, W. Li, "A zero-knowledge-proof-based digital identity management scheme in blockchain," *Computers & Security*, vol.99, 2020. Article (CrossRefLink)
[18] D. Chaum, "Blind Signature System," in *Proc. of Advances in Cryptology: Proceedings of Crypto 83*, 1984. Article (CrossRefLink)

[19] M. B. M. Kamel et al., "Attribute Verifier for Internet of Things," in *Proc. of 2022 32nd International Telecommunication Networks and Applications Conference (ITNAC)*, pp.1-3, 2022. Article (CrossRefLink)
[20] B. Parno et al., "Pinocchio: nearly practical verifiable computation," *Communications of the ACM*, vol.59, no.2, pp.103-112, 2016. Article (CrossRefLink)
[21] S. Chatterjee et al., "On the Efficiency and Security of Pairing-Based Protocols in the Type 1 and Type 4 Settings," in *Proc. of Arithmetic of Finite Fields, Third International Workshop, WAIFI 2010*, vol.6087, pp.114-134, 2010. Article (CrossRefLink)

Geunyoung Kim is a PH.D. student in Department of Computer Engineering at Chungnam National University in Republic of Korea. He received the B.S. degree in Computer Science & Engineering from Chungnam National University in 2012. He is interested in most aspects of information security and particularly focused on conducting research in the fields of blockchain and user authentication in web3.

Yunsik Ham is a B.S student in Department of Computer Engineering at Chungnam National University in Republic of Korea. He is interested in consensus algorithms in blockchain and various platforms.

JaeCheol Ryou is a professor in the Department of Computer Engineering at Chungnam National University in Korea. He received the B.S. degree in Industrial Engineering from Hanyang University in 1985, the M.S. degree in Computer Science from Iowa State University in 1988, and the Ph.D. degree in Electrical Engineering and Computer Science from Northwestern University in 1990. His research interests are Internet Security and Electronic Payment Systems.