I See You Blockchain User, or Not!
Privacy in the Age of Blockchains

Ghada Almashaqbeh

University of Connecticut
ghada.almashaqabeh@uconn.edu

Abstract

Cryptocurrencies and blockchains continue to build an innovative computation model that paved the way for a large variety of applications. Smart contracts made this even easier as they allow utilizing existing blockchain infrastructures to deploy new decentralized applications. However, privacy is a huge concern especially for permissionless public blockchains. We investigate this issue, explore current solutions and technology trends, define the gaps, and then discuss the requirements for a generic framework to bring privacy to smart contracts.

Main Takeaway: Preserving privacy of blockchain users is a complicated task, and usually relies on advanced cryptographic primitives. This impacted efficiency, usability, and understanding of the landscape. More effort is needed to close these gaps and devise provably secure tools to make it easier for system designers, developers and engineers, and even the end user, to asses and use blockchain-based privacy preserving systems.

1 Introduction

Following their revolutionary economic impact, cryptocurrencies and blockchain technology continue to build an innovative computation model. Researchers and practitioners alike are racing to build new applications and transform existing systems into fully decentralized ones by utilizing the unique features of blockchains. While early systems focused mainly on payment transfers, newer smart contract-enabled blockchains allow individuals to build applications that can process highly sensitive data, such as those involved in medical records tracking, trading, and voting.

However, lack of privacy is a huge concern. Popular systems like Bitcoin and Ethereum do not support privacy out of the box. All records are logged in the clear on the blockchain, allowing anyone to read their content. Moreover, while no real identities are required, several studies showed how the random-looking addresses can be linked to the real identities of their owners [11, 2]. This is problematic; users do not want their payment activity to be disclosed, not to mention more sensitive data such as their votes or health-related information. At the same time, revealing coins history may result in what is known as tainted currency. These are coins that no one wants to own (or accept as a payment) due to some undesired coin history (like being used in some illegal trade). A currency is supposed to be fungible in order to serve its basic purposes.

Such concerns resulted in several academic and industrial initiatives to bring privacy to blockchains [12, 15, 4, 3, 17, 18, 6, 9, 10]. These started with the simpler Bitcoin model and provided confidential payments (only hide transfer amount) and anonymous ones (hide user addresses). More recently, new directions emerged to build private smart contracts, allowing for arbitrary computations to be performed on the blockchain with private inputs/outputs, anonymous user identities, and even preserving function privacy in which the computation itself is hidden.

Nonetheless, privacy-preserving solutions face several challenges. They may introduce an efficiency bottleneck since they usually involve computationally heavy cryptographic operations, such as homomorphic commitments/encryption [15, 4] and various zero knowledge proof systems [14, 8, 5]. Moreover, they may pose difficulties in complying with regulations. That is, blockchain and cryptocurrencies introduce a new relation with finance cultivated with the term of DeFi (decentralized finance). This is basically building decentralized blockchain-based versions of conventional financial services, such as loan management and insurance claims, without relying on centralized banks. Privacy is a natural requirement here; no one wants their activity records to be public. But this

pegs the question of how to comply with known regulations that banks adhere to, such as know your customer policy (KYC), taxes, etc.?

Furthermore, and similar to any technology, privacy preserving solutions could be utilized by malicious attackers to hide their tracks. The Silkroad website [1] (a darknet drug market) used Bitcoin to handle payments with the thinking that it is fully anonymous. Such issues raise the question of whether we can develop powerful privacy preserving solutions that can authenticate users without disclosing their identities, and can track them if they misbehave. This adds to the complexity of the problem; addressing privacy in highly distributed environments such as blockchains is challenging, let alone handling compliance.

2 The Landscape

Privacy support in blockchain-based systems can be divided into four categories: transaction confidentiality, user anonymity, generalized privacy preserving computations, and function privacy

Transaction confidentiality hides the currency amount and the sender/recipient account balances by (usually) encrypting them. Anonymity hides the addresses of the sender and the recipient by using anonymity sets. That is, the sender/recipient could be anyone in a set of users, making it hard to guess. Confidentiality and anonymity are usually combined with each other to achieve full user privacy in currency transfer. Zerocash [15], Monero [13], and Quisquis [7] are examples of a system that achieves such capability.

The category of generalized privacy preserving computations is mainly related to smart contract-enabled systems. Such systems allow end users to write any program and deploy it on the blockchain, so the miners execute the code on their behalf. Thus, this category extends the first and second categories (namely, transaction confidentiality, user anonymity) to allow this public code to operate on private inputs and/or produce private outputs. Examples of systems under this category include Zether [4], which supports a limited set of operations, and smartFHE [16] that can support any user application.

The fourth category takes this even further to support function privacy; not only the inputs and outputs are hidden but also the program code itself. This could be of particular interest for proprietary code, or when the code may leak the type of processed inputs (e.g., votes) which may have privacy implications. Zexe [3] is a system that provides this privacy type, but it targets Bitcoin-like systems rather than smart contract-enabled ones.

The main ingredients of these solutions are centered around a handful of cryptographic primitives. They followed a similar paradigm of hiding (mainly encrypting) inputs, performing the intended computation to produce hidden (encrypted) outputs, and proving computation correctness without revealing any private information. The primitives used include homomorphic encryption (commitments can be used, but for the interest of space we do not discuss that here), and zero knowledge proofs. In detail, for the case of currency transfers, currency amount and user balances are encrypted. Then receiving or sending currency corresponds to addition/subtraction operations over encrypted account balances. This is possible due to the use of additive homomorphic encryption (i.e., adding two encrypted inputs produces an encryption of the sum of the actual inputs, and same for subtraction). Validity of operations is attested for using zero knowledge proofs, which provide cryptographically irrefutable proofs that all agreed upon conditions are met without revealing anything about the hidden data. If fully homomorphic encryption is used, then arbitrary computations can be performed on encrypted inputs.

For function privacy, on the other hand, current solutions let the users perform the computation locally on their private data, then post only the encrypted output with a proof attesting to the correctness of the computation. Thus, no one can tell what function was applied nor the actual values of the inputs/outputs.

Why is it harder for smart contracts? First, a smart contract can be any program of the user's choice that may involve complex operations. Supporting privacy in this case is more challenging than (the simpler) basic currency transfer operations. Second, smart contracts may involve inputs coming from different users, meaning that these inputs are encrypted under different keys. Allowing such interoperation is not trivial and usually requires sophisticated cryptographic primitives. Lastly, the flexibility provided by smart contracts raises several questions related to correctness and legitimacy of the deployed code. What if this code simply reveals all inputs and/or users addresses? This places a huge responsibility on the end user to vet such applications and contracts before using them.

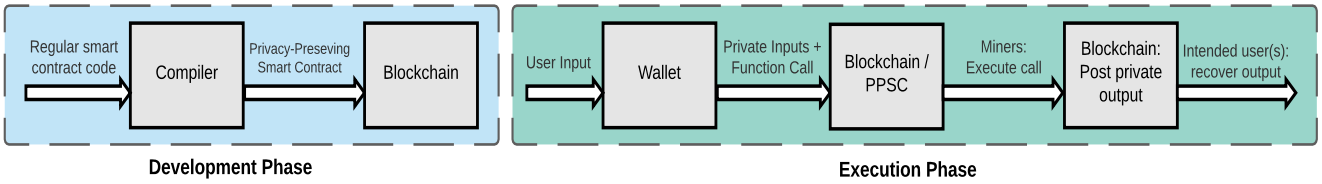

Figure 1: A generic framework for privacy preserving smart contracts.

2.1 A Path Towards A Usable Private Smart Contract Framework

A smart contract is a program containing code that implements a specific application. The code and its state are recorded on the public blockchain, and anyone can invoke any function within this code. Our goal is to explore the requirements for a framework that allows building privacy preserving smart contracts. That is, one that allows operating on private inputs, producing private outputs, maintaining a private state, but the code itself is still public since we want the miners to execute the code for the users. We envision a framework with a structure as shown in Figure 1.

Compiler. Supporting privacy means that regular smart contract code (that operates on public inputs) needs to be converted to a format that operates on private inputs by using proper cryptographic primitives. For usability purposes (to allow developers who do not have deep cryptography knowledge to use this framework), this conversion can be automated by having compilers that convert regular contracts into privacy-preserving ones.

User input. As anyone can ask the miners to execute the smart contract code by sending the proper inputs, it is the user's responsibility to prepare the right format of the private input. Handling this task can be also automated for the end user through integrating this capability into their cryptocurrency wallets. A wallet is the software a user installs to interact with a blockchain-based system and track the coins this user owns. The wallet can be extended to have two modes, public and private, and based on the type of the smart contract a user wants to invoke, the wallet can produce the right input format (if private, it will encrypt the input, if public, the input will stay as is).

Code execution. Step one (compilation) produces the suitable code for the miners to execute and produce private outputs. A question that arises here is regarding the cost of this execution especially with the heavy cryptographic primitives involved. If these primitives are executed at the smart contract level, i.e., as part of the application layer, then the miners' fees (e.g., gas fees in Ethereum) will be extremely high. An alternative solution is to make these primitives part of the instruction set that the system implements natively (e.g., as precompiles in Ethereum's virtual machine). This will greatly reduce the cost and make debugging easier since it forces all users to use the same primitives.

Output dissemination. The beauty of blockchain-based systems is that they do not require end users to be always online. A user can invoke a specific function and then go offline, come back later to check the output which the miners will record on the blockchain. The output will be encrypted and only the intended user will be able to decrypt and read its value. If multiple users are involved (i.e., the output was based on their inputs), then all these users need to coordinate with each other to run a protocol to decrypt the output.

Practical viability. Several issues face such generic framework on both the design and implementation fronts. Handling concurrency is challenging. That is, an account state (like its balance) must not change so long as there is a pending private operation tied to its current state, otherwise, this operation will be invalidated (since the zero knowledge proofs will become invalid if state changes). Another concern is efficiency as mentioned before; luckily, the field is witnessing rapid advancements and several major players are paying attention to privacy preserving technologies. For example, Microsoft has its own library, called Seal, that looks into optimizing the implementation of fully homomorphic encryption. We expect to see highly efficient practical deployments similar to what happened in the field of zero knowledge proofs.

References

[1] Silk road marketplace. https://en.wikipedia.org/wiki/Silk_Road_(marketplace).

[2] Alex Biryukov and Sergei Tikhomirov. Deanonymization and linkability of cryptocurrency transactions based on network analysis. In *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 172–184. IEEE, 2019.

[3] Sean Bowe, Alessandro Chiesa, Matthew Green, Ian Miers, Pratyush Mishra, and Howard Wu. Zexe: Enabling decentralized private computation. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 947–964. IEEE, 2020.

[4] Benedikt Bünz, Shashank Agrawal, Mahdi Zamani, and Dan Boneh. Zether: Towards privacy in a smart contract world. In *International Conference on Financial Cryptography and Data Security*, pages 423–443. Springer, 2020.

[5] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 315–334. IEEE, 2018.

[6] Raymond Cheng, Fan Zhang, Jernej Kos, Warren He, Nicholas Hynes, Noah Johnson, Ari Juels, Andrew Miller, and Dawn Song. Ekiden: A platform for confidentiality-preserving, trustworthy, and performant smart contracts. In *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 185–200. IEEE, 2019.

[7] Prastudy Fauzi, Sarah Meiklejohn, Rebekah Mercer, and Claudio Orlandi. Quisquis: A new design for anonymous cryptocurrencies. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 649–678. Springer, 2019.

[8] Jens Groth. On the size of pairing-based non-interactive arguments. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 305–326. Springer, 2016.

[9] Harry Kalodner, Steven Goldfeder, Xiaoqi Chen, S Matthew Weinberg, and Edward W Felten. Arbitrum: Scalable, private smart contracts. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*, pages 1353–1370, 2018.

[10] Thomas Kerber, Aggelos Kiayias, and Markulf Kohlweiss. Kachina-foundations of private smart contracts. *IACR Cryptol. ePrint Arch.*, 2020:543, 2020.

[11] Philip Koshy, Diana Koshy, and Patrick McDaniel. An analysis of anonymity in bitcoin using p2p network traffic. In *International Conference on Financial Cryptography and Data Security*, pages 469–485. Springer, 2014.

[12] Ian Miers, Christina Garman, Matthew Green, and Aviel D Rubin. Zerocoin: Anonymous distributed e-cash from bitcoin. In *2013 IEEE Symposium on Security and Privacy*, pages 397–411. IEEE, 2013.

[13] Shen Noether, Adam Mackenzie, et al. Ring confidential transactions. *Ledger*, 1:1–18, 2016.

[14] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In *2013 IEEE Symposium on Security and Privacy*, pages 238–252. IEEE, 2013.

[15] Eli Ben Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 459–474. IEEE, 2014.

[16] Ravital Solomon and Ghada Almashaqbeh. smartfhe: Privacy-preserving smart contracts from fully homomorphic encryption.

[17] Samuel Steffen, Benjamin Bichsel, Mario Gersbach, Noa Melchior, Petar Tsankov, and Martin Vechev. zkay: Specifying and enforcing data privacy in smart contracts. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 1759–1776, 2019.

[18] Guy Zyskind, Oz Nathan, et al. Decentralizing privacy: Using blockchain to protect personal data. In *2015 IEEE Security and Privacy Workshops*, pages 180–184. IEEE, 2015.