Towards a quantum-safe transaction signature in Ethereum

Stefano Bistarelli[1,†], Marco Fiore[2,*,†], Ivan Mercanti[1,†] and Marina Mongiello[2,†]

[1]*Dipartimento di Matematica e Informatica, University of Perugia, Perugia, Italy*
[2]*Dipartimento di Ingegneria Elettrica e dell'Informazione, Politecnico di Bari, Bari, Italy*

Abstract
In order to maintain security, safeguard user privacy, and improve system performance, blockchain systems mainly rely on cryptographic techniques. Ethereum, for instance, controls currency ownership using the Elliptic Curve Digital Signature Algorithm (ECDSA), ensuring that only legitimate money owners may use it. However, growing concerns are being raised about how resilient implemented cryptographic algorithms will be to quantum attackers due to the advent of quantum computing. As a result, the previous computational hardness assumptions may no longer be valid. The key-pair generation and transaction signature are the blockchain parts that most need a quantum-safe approach. We provide an interface implementation of an Ethereum node for supporting different signature methods. Moreover, the interface presents functions to generate a key-pair, sign and verify transactions, extract a public key from the private one, and get an address from the corresponding public key.

Keywords
Ethereum, Post Quantum, Blockchain cryptography, Signature

1. Introduction

Blockchain systems depend heavily on cryptographic methods to ensure security, protect user privacy, and enhance system performance. One of the essential cryptographic primitives for blockchains is the digital signature. For instance, Bitcoin [1] or Ethereum [2] uses the Elliptic Curve Digital Signature Algorithm (ECDSA) to control currency ownership, ensuring that only the rightful money owners can use it. A traditional signature system, in particular, consists of a private signing key and a public verification key. A 160-bit hash of a bitcoin address is created deterministically using the ECDSA verification key. The right to use the money represented by the Bitcoin address belongs to anyone with the associated ECDSA signature key. Although the importance of cryptographic primitives is widely acknowledged, the development of quantum computing prompts growing worries about how resilient implemented cryptographic algorithms will be to quantum adversaries because the computational hardness assumptions that have been in place in the past may no longer be valid[1]. Furthermore, the recent claims of "quantum supremacy" made by Googleand IBM[2] have only served to increase the demand for post-quantum secure blockchains. So, in order to ensure the future dependability of blockchain technologies, it is crucial to investigate the possible applications of quantum computing to weaken and strengthen blockchain technologies.

5th Distributed Ledger Technology Workshop (DLT 2023)
*Corresponding author.
†These authors contributed equally.
stefano.bistarelli@unipg.it (S. Bistarelli); marco.fiore@poliba.it (M. Fiore); ivan.mercanti@unipg.it (I. Mercanti); marina.mongiello@poliba.it (M. Mongiello)
0000-0001-7411-9678 (S. Bistarelli); 0000-0003-0102-2394 (M. Fiore); 0000-0002-9774-1600 (I. Mercanti); 0000-0002-1477-1434 (M. Mongiello)

[1]https://www2.deloitte.com/nl/nl/pages/innovatie/artikelen/quantum-computers-and-the-bitcoin-blockchain.html. .html.
[2]https://www.ibm.com/blogs/research/2019/10/on-quantum-supremacy/.

Three main components of a Blockchain architecture can each use a quantum-safe approach: i) Key-pair generation: a post-quantum algorithm can quickly generate a user's private key starting from his public key; ii) Transaction signature: a post-quantum algorithm can quickly generate a user's private key starting from the signature of a transaction; iii) Block hashes: a post-quantum algorithm can be used by an attacker to break the hash linking between blocks, writing new data in a block and quickly rebuilding the entire chain. Our effort is concentrated on themes i) and ii). Due to the distributed nature of the Blockchain, the third subject is not regarded as a serious issue: if a single user rebuilds the hash linking, he will have a functional chain. However, his chain will not be part of the main net since other nodes will not accept his update. We provide an interface for supporting different signature methods on the Go Ethereum node implementation[3].

2. Changes in GO code

The Geth implementation is the most used one, so we decided to focus on its version v1.11.0 (Annos Basin), to analyze the code and develop our approach to support a quantum safe signature algorithm. We identify different paths and files to modify to switch from the use of ECDSA signature algorithm to multiple algorithms. The main folder to analyze is located in *go-ethereum/crypto*: in particular, we analyze the *crypto.go* and *signature_cgo.go* files.

To parameterize the methods contained in those files, we created *crypto_ecdsa.go* and *signature_cgo_ecdsa.go*. Each time a method is called in the go-ethereum code, the calling functions make a check on the used algorithm and call the right function with its inputs and expected outputs. The main constraint is to use an ECDSA-compliant algorithm, that is, an algorithm using the same dimensionalities (i.e., public and private key size, signature size, signature parameters) and formats of ECDSA. To overcome this issue, we plan to improve our interface further, considering only the main methods and making them accept, as input and output, byte-formatted parameters. Parsing functions will then be used to adapt the parameters to each used signature algorithm.

A node has been run and tested using the interface. The geth console is operational and correctly runs after generating a key-pair.

3. Conclusions

This work presents an interface implementation of an Ethereum node for supporting different signature methods to preserve post-quantum resistance. We modified the Go Ethereum node implementation because it is the most used one in the implementations and the first to be updated with new features.

First, we determine which files and directories need to be changed to convert from using the ECDSA signature technique to various algorithms. Then, we modify the ECDSA methods making them parametric. Moreover, we made two new libraries to parameterize such methods. Finally, we ran and tested the new node with the interface showing that after creating a key-pair, the geth console is active and functions as intended.

In the future, we plan to extend our interface in order to use several quantum resistance signature algorithms. In particular, we would like to start with SPHINCS [3]. An interface that supports different signature methods should present functions to generate a key-pair, sign and verify transactions, extract a public key from the private one, and get an address from the corresponding public key. To ensure that different signature algorithms works in such environment, public and private keys should be used in byte format, so a parsing function should be called before and after the interface methods. We plan to let nodes choose the algorithm during the Geth initialization phase, using a flag to identify the desired cryptographic algorithm.

[3]https://geth.ethereum.org/.

References

[1] S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system, Bitcoin project white paper (2009).

[2] V. Buterin, Ethereum white paper: A next generation smart contract & decentralized application platform (2013). URL: https://github.com/ethereum/wiki/wiki/White-Paper.

[3] D. J. Bernstein, D. Hopwood, A. Hülsing, T. Lange, R. Niederhagen, L. Papachristodoulou, P. Schwabe, Z. Wilcox-O'Hearn, SPHINCS: practical stateless hash-based signatures, IACR Cryptol. ePrint Arch. (2014) 795. URL: http://eprint.iacr.org/2014/795.