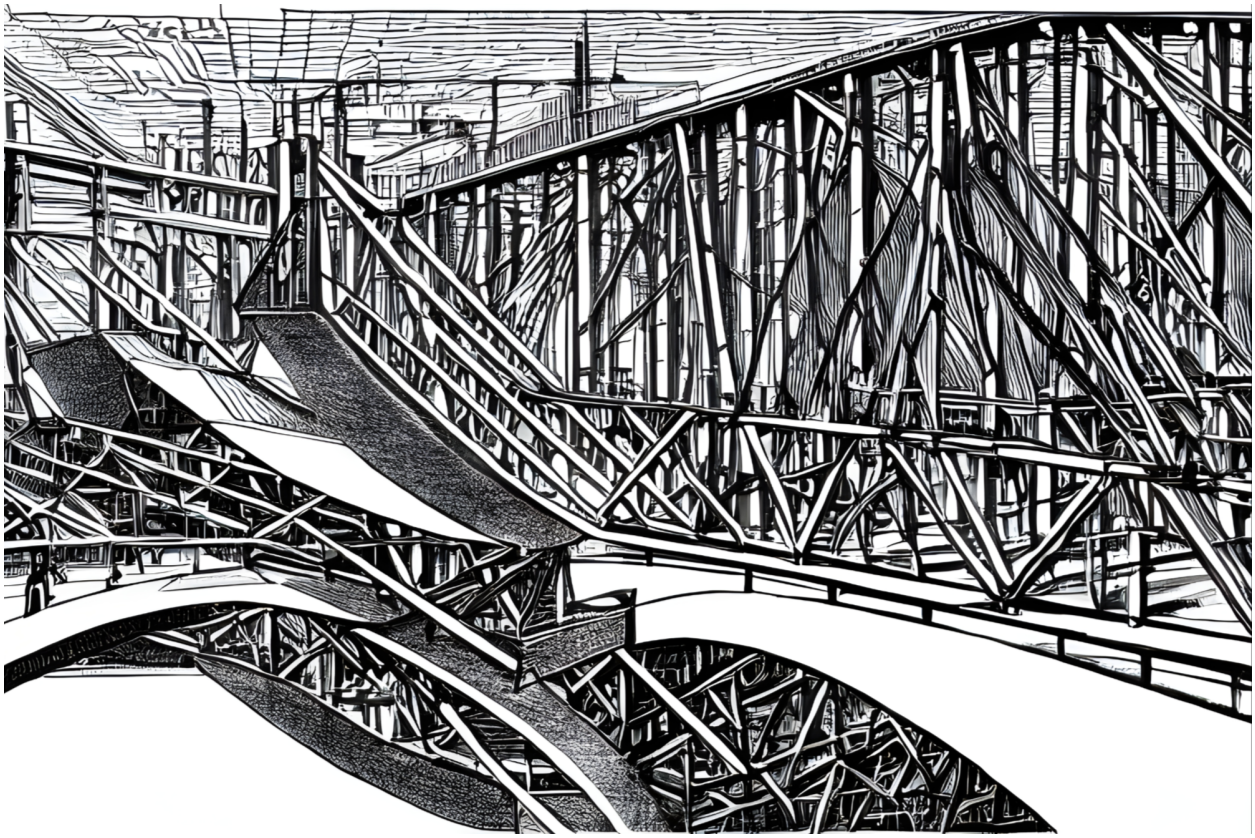
Yellow Paper (May 2023, Version 1.0)

Post-Quantum Zero-Knowledge (PQZK) Bridge

Abelian Foundation



Abstract

We propose and outline an idea to build an inter-blockchain connectivity bridge between the post-quantum Abelian Blockchain and a conventional non-quantum-proof Blockchain. This bridge makes use of the quantum resistance property of Abelian to upgrade wallets and digital assets on the conventional Blockchain system and make them secure against quantum attacks. We call this bridge a Post-Quantum Zero Knowledge Bridge (PQZK Bridge).

Table of Contents

1. Introduction

The rise of quantum computing poses a significant threat to the security of traditional cryptography, which is widely used to secure data, communication, and transactions on the internet. As quantum computers can solve complex mathematical problems, they could potentially be used for breaking cryptographic algorithms, especially those elliptic curve based digital signature schemes such as ECDSA and Ed25519 that underlie almost all the Blockchain systems of today.

The security of an elliptic curve based cryptographic algorithm is based on the assumption that solving a discrete logarithm problem or its variants under some elliptic curve groups is intractable. However, these hard mathematical problem assumptions are no longer valid against quantum computing.

In 1994, Shor proposed a quantum algorithm that can break the set of elliptic curve hard problems efficiently. This implies that once quantum computers become powerful enough (with adequate number of qubits as a prerequisite), we can make use of such a quantum computer and run the Shor's quantum algorithm to break a system whose security relies on an elliptic curve based cryptosystem.

Not only this concern applies to layer 1 blockchain systems such as Bitcoin and Ethereum, but is also a pressing issue to all other elliptic curve based cryptosystems including those bilinear pairing based zero knowledge proof systems, for example, the Zero-Knowledge Succinct Non-interactive Argument of Knowledge (zk-SNARK).

To address this industry-wide security issue, in this yellow paper, we propose and outline an idea about building an inter-blockchain connectivity bridge, that we call a Post-Quantum Zero-Knowledge Bridge (**PQZK Bridge**). In particular, we devise a zero knowledge proof system based inter-blockchain connectivity bridge which connects the post-quantum Blockchain system, Abelian, with a conventional non-quantum-proof Blockchain system, such as Bitcoin or Ethereum. This bridge then allows wallets, smart contracts and digital assets on this conventional Blockchain system to become quantum-proof.

The PQZK Bridge leverages the quantum resistance property of the Abelian Blockchain and upgrades wallets on a conventional Blockchain system to quantum-proof. This will help users continue enjoying all the great features (such as scalability) and dApps on the conventional Blockchain systems while at the same time ensure that their digital assets on those Blockchain systems will be secure against quantum attacks.

For instance, BTCs on Bitcoin and ETHs or any digital assets or smart contracts on Ethereum are not quantum-resistant. The ownership of digital assets on these Blockchain systems is not assured in the long run and will readily be compromised once an attacker possesses a powerful quantum computer, especially at the early stage of the quantum computing development, this quantum breaking machine could be a special-purpose one.

The techniques and implementation of the PQZK Bridge may also have a wider implication to the blockchain industry as a whole. Post-quantum cryptography and particularly the zero-knowledge proof systems that we have built on the Abelian Blockchain and will be built for this PQZK Bridge could potentially be applied to a much wider range of use cases.

As always, we make the design and code open source, and encourage scholars and developers to participate, and make use of all the tools we built on their projects. We hope that this yellow paper can serve as a timely contribution to the field of post-quantum blockchain security, and attract more contributions from the community to this very important topic as the world has already started migrating to post-quantum cryptography.

2. Blockchain Insecurity Against Quantum Computers

Quantum computers are a new class of computing devices that use quantum bits (qubits) instead of classical bits to perform computations. Quantum computers have the potential to solve certain problems much faster than classical computers, including breaking cryptosystems used in current Blockchain systems, for example, the elliptic curve cryptography.

There are two quantum algorithms that could threaten the security of a conventional Blockchain system: Shor's algorithm and Grover's algorithm. Shor's algorithm can efficiently factor large integers and its variant is able to solve discrete logarithm problems, and hence their variants (e.g. computational / decisional Diffie-Hellman problem) efficiently, while Grover's algorithm can search an unstructured database of N items in O(N^(1/2)) time, and hence can be used for speeding up the brute-force attacks against hash functions and symmetric-key encryption algorithms in certain extent.

The time complexity of Shor's quantum algorithm for factoring an integer N, where N is the number of bits in the key, is approximately $O((\log N)^3)$, which is significantly faster than the best-known classical algorithms for factoring large integers. In a conventional Blockchain system, for example in Bitcoin or Ethereum, when a transaction is initiated, the public key is made public. A quantum attacker can then compromise the public key and recover the corresponding secret key using Shor’s quantum algorithm. This would entail catastrophic losses. For example, an attacker can transfer assets from vulnerable accounts by examining states stored in nodes to compute the private keys from the public keys. Or an attacker could initiate another transaction, transfer the assets to his own address before other

miners process the original transaction, and raise the mining fee to ensure that his new transaction is verified before the original transaction.

On the other hand, Grover's algorithm can speed up the process of finding a preimage of a given hash value in an unsorted database of N items in O(N^(1/2)) time, which is faster than the best-known classical algorithms for this problem. However, this speedup is only a modest improvement over classical algorithms, and it does not pose a significant threat to the security of most commonly used hash functions.

As a summary, we can see that by using Shor's algorithm, a quantum adversary can compute the private key from a public key which has a significant impact on conventional non-quantum-proof Blockchain systems. However, Grover's algorithm has no significant effect on Blockchain's security as long as we double the security parameter, for example, increasing the hash function output range from 256 bits to 512 bits.

Quantum computing research is advancing rapidly, and the National Institute of Standards and Technology (NIST) [6] and many other countries are actively working on developing post-quantum standards and transitioning to post-quantum environments. It is essential to consider post-quantum security for all Blockchain systems while the world is migrating towards post-quantum cryptography.

3. A Brief Introduction to Abelian, a Post-Quantum Layer 1

Abelian is a Layer 1 Blockchain system which is quantum proof and privacy preserving. By quantum proof, Abelian's cryptographic systems can defend against attacks launched by the future powerful quantum computers. By privacy preserving, Abelian has the privacy coin ABEL, which can be used with full privacy, privacy with accountability, or privacy in the pseudonymous level.

This is the world's first Blockchain system which is both post-quantum and privacy-preserving. Abelian's lattice-based post-quantum cryptosystems are based on CRYSTALS-Dilithium and Kyber, both have been chosen by the US National Institute of

Standards and Technology (NIST) as in the first batch of post-quantum cryptographic (PQcrypto) national standards. This symbolizes the mass adoption of these PQcrypto primitives by the federal agencies and the rest of the world for replacing conventional elliptic curve cryptosystems.

Abelian makes use of linkable ring signatures in the context of post-quantum cryptography to achieve privacy-preserving. While different from Monero, Abelian has multiple levels of privacy-preserving. It spans from pseudonymity to full privacy. Pseudonymity gives similar anonymity assurance to that by Bitcoin. Full privacy protects sender and receiver wallet addresses, and also the transaction amount. In between pseudonymity and full privacy, Abelian also supports accountability, which lets a user share a trapdoor information with a third party in such a way that full privacy is still ensured against the public while it becomes pseudonymous to this third party.

This post-quantum Blockchain project started 5 years ago in 2018. The founding team of Abelian consisted of cryptographers and academic researchers who had the vision of building a post-quantum Blockchain system which also supports multiple levels of privacy preserving. A lattice-based linkable ring signature was later proposed by the Abelian team. It is currently used to ensure the privacy of the sender of a transaction on Abelian. Some lattice-based commitment scheme and zero knowledge proof system have also been employed together with a lattice-based encryption algorithm with key privacy feature, for example, the CRYSTALS-Kyber, being employed for building up this post-quantum privacy-preserving Abelian Blockchain system.

For more details on the technical details of Abelian, please refer to the whitepaper, https://www.abelian.info/whitepaper.pdf, and other writeups available at https://pqabelian.medium.com.

4. Upgrading to Quantum-Safe

We now propose an inter-chain connectivity bridge between a conventional non-quantum-proof Blockchain and Abelian, which is quantum-proof. The bridge will let individual users on the conventional Blockchain bind their wallets to quantum-proof wallets om Abelian first. Then the bridge will enable users to make their execution of smart contracts and their digital assets on the conventional Blockchain quantum safe.

We aim at making our techniques as generic as possible so that the same suite of techniques can be extended to other Blockchain systems. While reading through our method described below, we may consider the underlying system of the conventional non-quantum-proof Blockchain as Bitcoin, Ethereum or any other major Blockchain systems with a comparable architecture. One may consider a conventional Blockchain system which is based on elliptic curve cryptography, in particular, on the curve secp256k1,

and its signature scheme is based on Elliptic Curve Digital Signature Algorithm (ECDSA) or its variants.

In the following, we use the terms quantum-safe, quantum-resistant, and quantum-proof interchangeably.

4.1 Quantum-Safe Wallets

Suppose Blockchain A is a conventional, non-quantum-safe blockchain, where the key pair of each wallet is based on the elliptic curve cryptography. Suppose Blockchain B is a quantum-proof blockchain, for example, Abelian. Let PK_A ← KGen(SK_A) be the digital signature key generation algorithm of Blockchain A. (PK_A, SK_A) represents the public-private key pair.

Suppose Alice would like to generate a wallet on Blockchain A and make use of Blockchain B to upgrade all transactions of Alice's wallet on Blockchain A quantum-safe.

To do so, Alice first generates a post-quantum public-private key pair (pqPK_A, pqSK_A), and the corresponding wallet address pqAddr_A from pqPK_A on Blockchain B. Let

pqPK_A ← pqKGen(pqSK_A)

be the post-quantum key generation algorithm of Blockchain B.

Next, Alice randomly generates a seed Seed_A and applies a quantum-safe one-way function pqH onto Seed_A to produce a private key SK_A for Blockchain A, namely

SK_A ← pqH(Seed_A).

Next, Alice computes

PK_A ←KGen(SK_A)

and applies the wallet generation hash function of Blockchain A on PK_A for generating the corresponding wallet address Addr_A. Remark: normally, the wallet generation hash function of Blockchain A for generating Addr_A from PK_A is already quantum-safe, and the

Grover's algorithm cannot compromise the hash function if its range is large enough, for example, at least 512-bit long. While in our construction, we do NOT require the Blockchain A wallet generation hash function to be quantum-safe.

4.1.1 Remarks on pqH

The domain of the quantum-safe one-way function pqH is the collection of any binary strings with the length of k bits, where k is the security parameter. This quantum-safe one-way function pqH maps a k-bit long random binary string (e.g. Seed_A) to the domain of the private key of a Blockchain A wallet.

For example, if Blockchain A uses ECDSA, the domain of a private key on Blockchain A is the interval of [1, n-1], where n is the order of the underlying elliptic curve group of the ECDSA, for example, on the curve of secp256k1, and n is usually 256 bits long. Also the length of the binary random string input (i.e. Seed_A) of pqH can be 256 bits long.

4.1.2 Remarks on Existing Key Pairs on Blockchain A

To make a wallet on Blockchain A quantum safe, Alice starts the journey by generating a brand new set of (PK_A, SK_A, Addr_A) as above. If Alice already has wallets on Blockchain A, we do not attempt to upgrade them to quantum safe as the solution does not seem to be elegant and it might increase the complexity of the entire quantum-safe upgrade process. We suggest not to consider all the complicated scenarios that may come across if we attempt to upgrade existing wallets and their corresponding key pairs to be quantum-safe. However, the question is how we can protect existing digital assets that Alice is currently holding with her existing wallets.

In fact, in most cases, the solution is quite straightforward. By simply transferring digital assets from an existing non-quantum-safe wallet on Blockchain A to a newly generated quantum-safe wallet (except those are non-transferrable such as a soulbound tokens where new soulbound tokens might need to be recreated), and from that moment on, digital assets which have already been transferred to the quantum-safe wallet will remain

quantum safe as long as they are transferred to only other quantum-safe wallets when needed to be transferred further. One should also notice that this transfer from non-quantum-safe wallet to a quantum-safe wallet has to be carried out well before the presence of a quantum adversary so that the transactions of the transfers can be well written on the ledger of Blockchain A.

4.2 Binding the Two Wallets Between Blockchains A and B

After generating the tuple <Seed_A, SK_A, PK_A, Addr_A, pqSK_A, pqPK_A, pqAddr_A>, Alice records the binding of the tuple on the Blockchain B ledger as follows.

First, Alice uses a post-quantum Non-Interactive Zero-Knowledge (NIZK) proof system pqZKP to generate a proof:

$$\text{Proof\_A} \leftarrow \text{pqZKP}\{ (\text{Seed\_A}, \text{r\_A}, \text{pqSK\_A}) : \text{PK\_A} \leftarrow \text{KGen}(\text{pqH}(\text{Seed\_A})) \wedge$$

$$\text{C\_A} \leftarrow \text{Com}(\text{Seed\_A}; \text{r\_A}) \wedge \text{pqPK\_A} \leftarrow \text{pqKGen}(\text{pqSK\_A}) \}$$

where Com is a post-quantum commitment scheme and r_A is a random string. A successful verification of Proof_A implies that the prover, namely Alice,

1. generates PK_A from Seed_A;
2. the committed value of C_A is the same; and
3. Generates pqPK_A from pqSK_A

while Alice does not leak any information regarding her secrets Seed_A and pqSK_A.

After generated proof_A, Alice stores a tuple (pqPK_A, PK_A, C_A, Proof_A) on the ledger of Blockchain B for verifiers (i.e. miners / validators) to verify the binding between these two wallets on Blockchain A and Blockchain B. Details on the verification will be given below when describing the technique which makes Alice's transactions on Blockchain A quantum proof (Section 4.3.2 Transactional Verification: (TxProof_A, Proof_A)).

4.2.1 Storing Wallet Binding Proof_A on Blockchain B Ledger

Regarding how to store the tuple (pqPK_A, PK_A, C_A, Proof_A) on the ledger of Blockchain B, for example, if this post-quantum Blockchain B is Abelian, Alice may create a UTXO transaction which sends 1 Neutrino (the smallest unit of an Abelian token, i.e. 1 ABEL = 10^7 Neutrinos) to herself, namely, setting the wallet addresses of both sender and receiver to her own: pqAddr_A.

The tuple (pqPK_A, PK_A, C_A, Proof_A) will then be stored in the metadata field of this transaction. As Alice initiates this UTXO transaction, the value of pqPK_A will also be in the transaction and hence in practice, we do not need to explicitly store pqPK_A in the metadata field.

In practice, Alice may create this post-quantum key pair (pqPK_A, pqSK_A) dedicated to the purpose of upgrading (PK_A, SK_A) on Blockchain A to quantum proof. Alice will create other key pairs, hence wallets, on Blockchain B for other transactions or purposes. This approach improves efficiency and enhances security.

It improves efficiency as all transactions on Blockchain B corresponding to (pqPK_A, pqSK_A) must be related to the quantum-safe upgrade to Alice's Blockchain A key pair (PK_A, SK_A). Hence some optimized data structure can be in place for Alice to handle this type of transactions efficiently. It also helps verifiers (i.e. miners / validators) retrieve needed transactions and metadata information efficiently.

It also enhances security as Alice can segregate the purposes of different key pairs and wallets on Blockchain B for more secure key / wallet management.

4.2.2 Verification of Wallet Binding Proof_A

Miners / validators on Blockchain B will verify Proof_A and write the tuple (pqPK_A, PK_A, C_A, Proof_A) onto Blockchain B's ledger. This is an extension to the verification of transactions on Blockchain B currently being carried out by miners / validators. If the verification of Proof_A passes, the miner / validator will then include the transaction and its corresponding metadata which contains the tuple (pqPK_A, PK_A, C_A, Proof_A) into a block.

In addition to the verification of Proof_A, in Section 4.3.2 Transactional Verification: (TxProof_A, Proof_A), we will see that Proof_A will be used side-by-side with a transaction level NIZK proof TxProof_A for ensuring that every single transaction of Alice on Blockchain A is quantum safe.

The Abelian Foundation is happy to make such an upgrade of Abelian for supporting the movement of upgrading all conventional Blockchain systems to quantum-proof. We believe that this upgrade is meaningful as we hope to support other blockchain ecosystems to be quantum-proof without making significant structural changes to them as the change of underlying cryptographic infrastructure of an existing Blockchain system can be quite challenging in many aspects.

For example, the key sizes and signature sizes of a post-quantum cryptosystem could be in the magnitude of hundreds of times larger than that of an elliptic curve cryptosystem. The scalability of a conventional Blockchain may no longer be effective and hundreds if not thousands of related systems such as wallets, validation software packages, communication protocols and many other components such as wallets need to be modified. Another concerning aspect is on security. Changing the cryptographic infrastructure of a conventional Blockchain could introduce new vulnerabilities. The system could become very complex and it could be very challenging to ensure changes on all related software components secure.

4.2.2 The Security of the Binding

From the tuple (pqPK_A, PK_A, C_A, Proof_A), where Proof_A is generated using the post-quantum NIZK proof system pqZKP, our purpose is to let verifiers / miners be able to ensure that the entity, namely Alice, who generated PK_A (so to Addr_A) is also the entity which generated pqPK_A or nobody else. We want to make sure that this proposition is true even in the presence of quantum adversaries.

From the post-quantum commitment scheme Com, the pqZKP ensures that the prover knows Seed_A which is committed to C_A, namely, $C_A \leftarrow Com(Seed_A; r_A)$. Furthermore, the prover has made use of the same Seed_A to generate $PK_A \leftarrow KGen(pqH(Seed_A))$

accordingly rather than resorting any other way which reverses the process from some point back to Seed_A given the value of PK_A. And the last part of Proof_A ensures that this same entity also generates pqPK_A from pqSK_A, namely, pqPK_A ← pqKGen(pqSK_A).

The table below summarizes the binding process of the two wallets from Blockchain A and Blockchain B.

Procedures for Creating and Binding the Two Wallets on Blockchain A and Blockchain B		
1.	pqPK_A ← pqKGen(pqSK_A)	pqPK_A will appear on Blockchain B (after the first transaction initiated by Alice)
2.	SK_A ← pqH(Seed_A), PK_A ←KGen(SK_A), Addr_A	PK_A and Addr_A will appear on Blockchain A
3.	Proof_A ← pqZKP{ (Seed_A, r_A, pqSK_A) :PK_A ← KGen(pqH(Seed_A)) ∧ C_A ← Com(Seed_A; r_A) ∧ pqPK_A ← pqKGen(pqSK_A) }	(PK_A, C_A, Proof_A) is written on Blockchain B

For implementation, we will develop a **PQZKBridge Wallet**. This PQZKBridge Wallet interacts with both Blockchain A and Blockchain B. It generates Seed_A, (PK_A, SK_A), Addr_A, (pqPK_A, pqSK_A), Proof_A and all related parameters. It also creates a transaction on Blockchain B for writing (PK_A, C_A, Proof_A) onto its ledger.

This PQZKBridge Wallet can be extended to integrate or connect with existing Blockchain A wallets (e.g. MetaMask) as well as existing Blockchain B wallets (e.g. Abelian Desktop wallet or abewallet) for more features, functionalities and enhancing user experience.

4.2.3 A Post-Quantum Signature on PK_A Would Not Work

We may wonder why we choose to construct a post-quantum NIZK proof system pqZKP for binding the two wallets between Blockchain A and Blockchain B. In fact, there is a more straightforward way to do so and it appears just fine intuitively.

This approach is to sign Alice's Blockchain A public key PK_A and/or wallet address Addr_A using the post-quantum digital signature scheme on Blockchain B under the post-quantum private key pqSK_A, namely,

$$pqSig \leftarrow pqSign(pqSK\_A, \text{“}PK\_A \Vert Addr\_A\text{”})$$

where pqSign represents the post-quantum digital signature generation algorithm on Blockchain B, for example, the CRYSTALS-Dilithium based signature scheme used on Abelian. pqSign takes as inputs the post-quantum private key pqSK_A and the message “PK_A ‖ Addr_A” where the symbol ‖ represents the binary string concatenation while “PK_A” and “Addr_A” are represented as the binary strings of Alice's public key and wallet address on Blockchain A.

After pqSig is generated, Alice stores pqSig in the metadata field of a transaction on Blockchain B.

For a quantum adversary, suppose the adversary has access to a powerful quantum computer, and can make use of the quantum computer to launch Shor's quantum algorithm to recover SK_A from PK_A. To reduce assumptions on the security of the solution, we also assume that the adversary is able to instantly recover SK_A from PK_A once PK_A becomes available on the ledger of Blockchain A.

Given the above outline of the adversarial model, we can see that it does not make much difference on protecting SK_A from being compromised by the adversary even if pqSig is a signature on “Addr_A” only rather than “PK_A”. This is because once Alice initiates a transaction on Blockchain A, her public key PK_A will be exposed on Blockchain A's ledger.

We can also see that given Addr_A or PK_A, and $pqSig \leftarrow pqSign(pqSK\_A, \text{“}PK\_A \Vert Addr\_A\text{”})$, the adversary can create his own quantum-safe public-private key pair on Blockchain B, say, (pqPK_A', pqSK_A') and creates another post-quantum signature:

$$pqSig' \leftarrow pqSign(pqSK\_A', \text{“}PK\_A \Vert Addr\_A\text{”})$$

and also stores pqSig' on Blockchain B. In this way, the adversary can abuse the mechanism by creating multiple key pairs on Blockchain B and generate multiple signatures pqSig”,

pqSig'",... on the same message "PK_A ‖ Addr_A", and write all these signatures onto Blockchain B.

This creates a problem that in practice, we also need a mechanism to choose and identify among all the potential post-quantum key pairs on which key pair is really Alice's. It turns out that this is not an obvious task.

If we set the rule that whichever digital signature pqSig is the first that appears on Blockchain B, we consider the corresponding (pqPK_A, pqSK_A) as Alice's legitimate post-quantum public-private key pair. However, this is inefficient as a miner / validator who needs to verify pqSig versus (pqPK_A, pqSK_A) has to scan through the entire Blockchain B ledger in sequence from the genesis block to the latest block for finding the first appearance of pqSig with respect to pqPK_A on the Blockchain B ledger. This action has to be done by each miner for every single transaction.

There will be a tradeoff between efficiency and storage requirement. If the miner wants to find out the legitimate (pqPK_A, pqSK_A) of Alice on Blockchain B efficiently, the miner may need to store the legitimate (pqPK_A, pqSK_A) for every single wallet on Blockchain B for quick lookup. This requires a lot of storage capacity as the number of distinct key pairs on Blockchain B continues to increase.

Another issue regarding this approach is that even with the rule above, namely, considering the first appearance of pqSig on Blockchain B refers to the legitimate post-quantum public key pqPK_A of Alice, it is not true that the corresponding pqPK_A has to be Alice's. In fact, an adversarial miner can intentionally delay the writing of the original pqSig and pqPK_A onto the ledger of Blockchain B. Instead, the adversary can intentionally delay the writing of pqSig under pqPK_A onto Blockchain B, while putting his own pqSig' under his own post-quantum public key pqPK_A'.

We are also not considering centralized approaches where Alice will solicit a centralized registration and bulletin service and put her (PK_A, Addr_A, pqSig) up there for publication.

As we can see, simply signing the Blockchain A public key PK_A or wallet Addr_A under Alice's post-quantum private key pqSK_A is insecure in the implementation described above.

4.3 Making Transactions on Blockchain A Quantum Proof

Now, after establishing a binding between (PK_A, SK_A) on Blockchain A and (pqPK_A, pqSK_A) on Blockchain B, we now describe how Alice can make her transactions on Blockchain A quantum proof.

Suppose Alice sends some native tokens (e.g. ETH) on Blockchain A. This transaction will contain a signature under (PK_A, SK_A). However, a quantum adversary is able to compromise SK_A from PK_A, which can be found on Blockchain A's ledger. The adversary can then generate a similar transaction with a valid signature under (PK_A, SK_A) and take ownership of all digital assets in the wallet Addr_A from Alice.

In order to make this transaction quantum proof, we can make use of Alice's knowledge on Seed_A, which cannot be compromised by a quantum adversary.

To do so, suppose BlockchainA_TxContent represents the content of the transaction. Suppose Pointer_A (e.g. a transaction hash and a block height on Blockchain B) is the information for retrieving the tuple (pqPK_A, PK_A, C_A, Proof_A) from Blockchain B ledger. Note that Alice readily knows Pointer_A when Alice generates the transaction on Blockchain B. Please refer to Section 4.2.1 Storing Proof_A on Blockchain B Ledger for details.

Alice computes

$$\text{TxProof\_A} \leftarrow \text{pqZKP}\{ (\text{Seed\_A}, \text{r\_A}) : \text{PK\_A} \leftarrow \text{KGen}(\text{pqH}(\text{Seed\_A})) \wedge \text{C\_A} \leftarrow \text{Com}(\text{Seed\_A}; \text{r\_A}) \}(\text{BlockchainA\_TxContent}, \text{Pointer\_A})$$

where (BlockchainA_TxContent, Pointer_A) represents the message part of the NIZK proof system pqZKP.

4.3.1 Transactional Proof Storage: TxProof_A

TxProof_A can be stored together with BlockchainA_TxContent on Blockchain A, for example, in the metadata field of the transaction BlockchainA_TxContent if this is supported.

Alternatively, similar to how we store Proof_A on Blockchain B as described in Section 4.2.1 Storing Proof_A on Blockchain B Ledger, TxProof_A can also be stored on the ledger of Blockchain B. To do so, Alice creates another transaction on Blockchain B to embed TxProof_A onto the transaction. This transaction can be a self-transfer of 1 Neutrino to herself. In this approach, there will be an additional parameter, pqPointer_A, which contains the location information of this transaction, for example, pqPointer_A = (hash of the transaction, block height) of the transaction written on Blockchain B's ledger.

There could be many other approaches in practice for storing TxProof_A so that it can be retrieved effectively. We will elaborate our actual implementation in the future.

4.3.2 Transactional Verification: (TxProof_A, Proof_A)

As described in Section 4.2.2 Verification of Wallet Binding Proof_A, miners / validators of Blockchain B have already verified and written the legitimate copy of Proof_A onto Blockchain B ledger. Also from Pointer_A, we can retrieve Proof_A for verification. There are three verifications that a miner / validator has to carry out for each transaction of Alice on Blockchain A:

1. The miner / validator of Blockchain A first carries out the conventional verification on BlockchainA_TxContent.
2. Then the miner / validator verifies Proof_A.
3. Finally, the miner / validator verifies TxProof_A.

The miner / validator prepares the transaction for writing (BlockchainA_TxContent, TxProof_A) on the ledger of Blockchain A if all the three verification steps above are passed. Note that as described in Section 4.3.1 Transactional Proof Storage: TxProof_A, TxProof_A may be stored on Blockchain B ledger instead. It may also potentially be stored somewhere else during the actual implementation.

4.3.3 Changes Needed on Blockchain A and Layer 2

As described above, Blockchain A's miners / validators need to additionally verify TxProof_A and Proof_A even if BlockchainA_TxContent is legitimate already. This is necessary as a quantum adversary is able to forge BlockchainA_TxContent which was not designed to be quantum-proof.

If miners / validators on Blockchain A are not upgraded for verifying TxProof_A and Proof_A, a quantum adversary can transfer digital assets sent to Alice's wallet Addr_A immediately on Blockchain A upon receiving those digital assets from other sources. Despite the quantum adversary is not able to generate a valid TxProof_A, if Blockchain A's miners / validators do not upgrade, hence will only verify the non-quantum-resistant part, BlockchainA_TxContent, but not TxProof_A or Proof_A, the quantum adversary can still be able to launch attacks and generate a valid BlockchainA_TxContent. Miners / validators on Blockchain A will continue writing BlockchainA_TxContent onto the ledger. Hence without upgrading miners / validators of Blockchain A, quantum adversaries can still transfer digital assets out from Alice's wallet Addr_A on Blockchain A.

We understand that it might be a bit challenging to require major Layer 1 conventional Blockchain systems to have the post-quantum verification add-ons deployed onto their consensus protocols. A more pragmatic approach might be developing a Layer 2 on Blockchain B while this Layer 2 behaves as a Sidechain of Blockchain A.

Let's call this Blockchain C, which not only allows the entire Blockchain community to get a much better understanding of this PQZK Bridge technology, but can also realize post-quantum smart contract capabilities much earlier.

To start the development of Blockchain C, we may clone the existing Blockchain A and then develop the storage mechanisms for TxProof_A and its auxiliary parameters such as Pointer_A, and also develop the verification of Proof_A and TxProof_A add-ons for Blockchain C's miners / validators. In the meantime, Blockchain B will also be upgraded by developing the corresponding storage mechanisms and the verification add-on for Proof_A.

Also, the PQZKBridge Wallet described in Section 4.4 PQZKBridge Wallet in Action will be developed.

Once Blockchain C becomes available, we can see the entire PQZKBridge technology in action on this Sidechain which has almost the identical resemblance to Blockchain A. For all patches and updates on Blockchain A, we can also port them over to Blockchain C. Imagine Blockchain A is Ethereum, and Blockchain C will therefore be a cloned Ethereum with the PQZK Bridge add-ons. It is also natural to continue building an ecosystem on top of this sidechain including a bridge connecting Ethereum with this PQZK Bridge enhanced Sidechain.

4.3.4 Scaling Up

It is feasible to improve the performance of this post-quantum transformation by aggregating multiple transactions of Alice into one transaction so that one post-quantum transactional proof TxProof_A can be generated for a set of transactions, namely

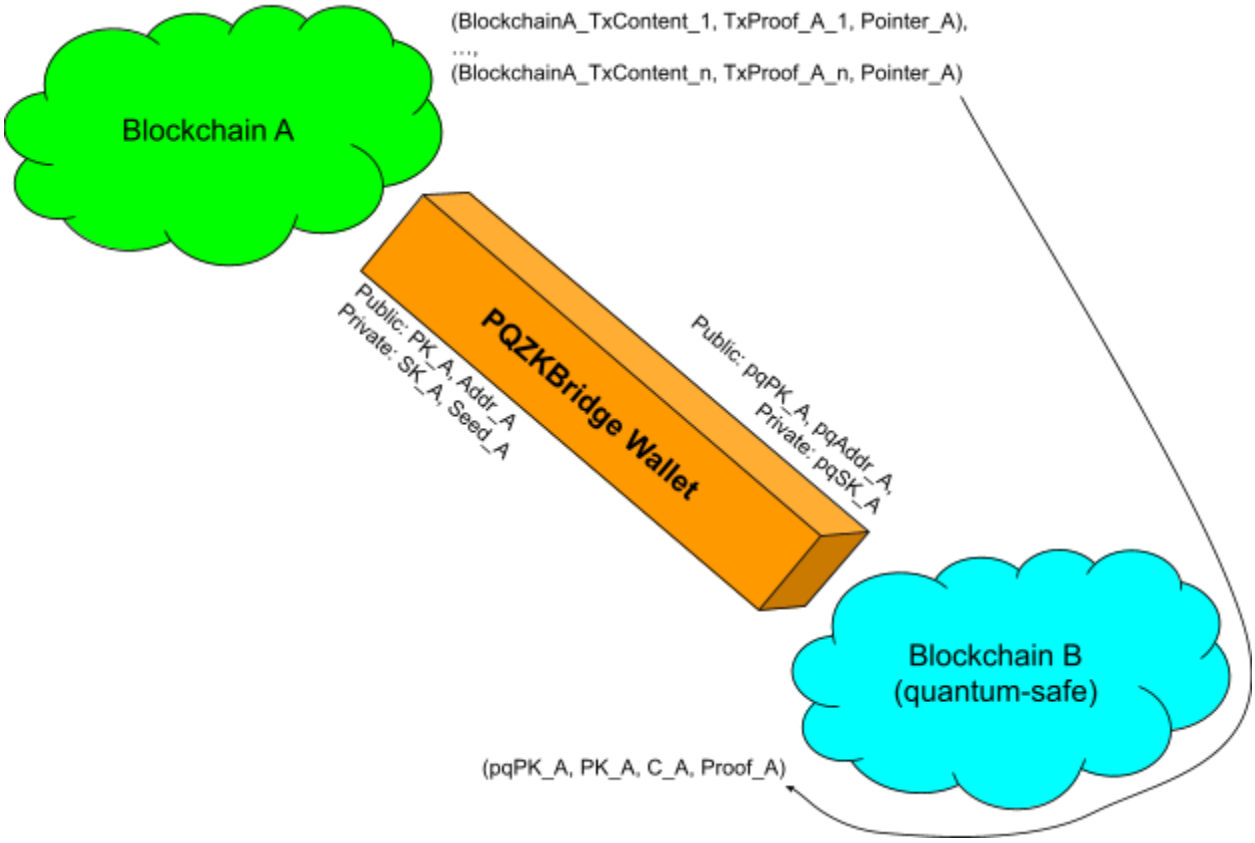
$$\text{TxProof\_A} \leftarrow \text{pqZKP}\{ (\text{Seed\_A}, \text{r\_A}, \text{pqSK\_A}) : \text{PK\_A} \leftarrow \text{KGen}(\text{pqH}(\text{Seed\_A})) \wedge \text{C\_A} \leftarrow \text{Com}(\text{Seed\_A}; \text{r\_A}) \}(\text{BlockchainA\_TxContent\_1}, \ldots, \text{BlockchainA\_TxContent\_n}, \text{Pointer\_A}).$$

Furthermore, it might also be possible to aggregate proofs from the transactions of different users for achieving even higher performance. This is analogous in concept to that of ZK-rollup.

We are also expecting the size of the proofs on ledger could be relatively large due to the actual post-quantum cryptosystem implementation. Being able to aggregate proofs will definitely help reduce the footprint of proofs on ledgers, we may also consider making proofs off-ledger as there is no need to verify all the old proofs once their corresponding transactions have been confirmed and deep in the ledger with much older block height. For example, we may remove all the off-ledger proofs if their corresponding transactions are more than 400,000 blocks old on Abelian.

4.4 PQZKBridge Wallet in Action

As mentioned in Section 4.2.2 The Security of the Binding, we will also implement a PQZKBridge Wallet which interacts with both Blockchain A and Blockchain B. In this section, we summarize the techniques above and walk through the steps that the PQZKBridge Wallet will carry out for Alice from the wallet creation to generating transactions on Blockchain A.



During the wallet creation, Alice makes use of the PQZKBridge Wallet to generate (pqPK_A, pqSK_A) and pqAddr_A. Next, (PK_A, SK_A) and Addr_A are also generated from Seed_A using PQZKBridge Wallet. The wallet then computes the tuple (pqPK_A, PK_A, C_A, Proof_A) and writes onto the ledger of Blockchain B indexed by Pointer_A.

When Alice is about to create a transaction on Blockchain A, Alice uses PQZKBridge Wallet to generate BlockchainA_TxContent_1 then computes TxProof_A_1. In general, for any transaction BlockchainA_TxContent_n created by Alice using PQZKBridge Wallet, the wallet also creates the corresponding transactional proof TxProof_A_n and writes it accordingly to the ledger.

PQZKBridge Wallet is also responsible for scanning through Blockchain A for discovering all transactions corresponding to Addr_A and carries out verification of the transaction's BlockchainA_TxContent, TxProof_A and Proof_A. PQZKBridge Wallet synchronizes up with both Blockchain A and Blockchain B for performing this task.

5. Conclusion

At Abelian Foundation, we hope to contribute to the Blockchain community with post-quantum cryptographic technologies which can help improve the quantum resistance of Blockchain systems. This yellow paper is considered as one of our first steps to achieve this. We leverage the quantum proof property of the Abelian Blockchain system to develop an inter-blockchain connectivity bridge which aims at upgrading other conventional Blockchain systems connected to the bridge to be quantum proof.

The PQZK Bridge idea we proposed here makes use of post-quantum NIZK proof systems to bind wallets of a conventional Blockchain system and the post-quantum Abelian Blockchain system together. Furthermore, the PQZK Bridge also upgrades individual transactions on the conventional Blockchain system to be quantum proof.

The techniques do require change on the mining / validating mechanism on the consensus protocol level, and may not be ideal for some Blockchain systems. However, when compared with other potential approaches such as upgrading the underlying cryptosystems of the conventional Blockchain systems to quantum-safe, the approach proposed in this paper seems to be much more straightforward, much less error prone, and much easier to implement in a modular fashion.

The changes described in this paper are add-ons which do not affect the existing functionalities of the conventional Blockchain systems. Our proposed mechanisms are add-ons which only require miners / validators carry out additional verifications on top of their existing verifications before adding transactions into blocks.

We hope that this modular approach is welcomed by most conventional Blockchain systems so that the Blockchain industry as a whole can get ourselves ready for the post-quantum era earlier.

– End –

Post-Quantum Zero-Knowledge Bridge (PQZK Bridge)

Revision History

May 2023: First Edition (version 1.0)