Committing to Quantum Resistance: A Slow Defence for Bitcoin against a Fast Quantum Computing Attack

I. Stewart[1], D. Ilie[1], A. Zamyatin[1,2], S. Werner[1], M.F. Torshizi, and W.J. Knottenbelt[1]

[1] Centre for Cryptocurrency Research and Engineering
Imperial College London, London, United Kingdom
[2] SBA Research, Vienna, Austria
{i.stewart, dragos.ilie14, a.zamyatin, sam.werner16, w.knottenbelt}@imperial.ac.uk
maryamtorshizy@gmail.com

Abstract. Quantum computers are expected to have a dramatic impact on numerous fields, due to their anticipated ability to solve classes of mathematical problems much more efficiently than their classical counterparts. This particularly applies to domains involving integer factorisation and discrete logarithms, such as public key cryptography.
In this paper we consider the threats a quantum-capable adversary could impose on Bitcoin, which currently uses the Elliptic Curve Digital Signature Algorithm (ECDSA) to sign transactions.
We then propose a simple but slow commit-delay-reveal protocol, which allows users to securely move their funds from old (non-quantum-resistant) outputs to those adhering to a quantum-resistant digital signature scheme. The transition protocol functions even if ECDSA has already been compromised. While our scheme requires modifications to the Bitcoin protocol, these can be implemented as a soft fork.

1 Introduction

Bitcoin is a decentralised digital currency system, which was introduced by the pseudonymous Satoshi Nakamoto in 2008 [32]. It leverages a peer-to-peer distributed network characterised by the lack of a central authority governing the state of transactions. Each consensus participant maintains a list of all historic transactions, grouped together in blocks, in a distributed public ledger called the blockchain. Blocks are chained together via the hashes of their predecessors, thereby providing strong guarantees for the immutability of the transaction history. Agreement on the current state of the system in the dynamically changing and pseudonymous set of participants is achieved by requiring nodes to solve complex cryptographic puzzles, known as Proof-of-Work (PoW). Consensus participants are known as miners and upon finding a valid solution to the PoW puzzle they are rewarded with new units of the underlying cryptocurrency and fees associated with the transactions included in the respective block.

Even though quantum computers (QCs) have theoretically existed for about 40 years, relatively recent breakthroughs placed the idea in the public eye once again. One such breakthrough, with a direct impact on Bitcoin's security, is Peter Shor's polynomial time quantum algorithm [45] that can, in its subsequently generalised form, break ECDSA. While more players enter this growing research area, it appears increasingly likely that powerful quantum computers will emerge in the near future. Although the early generations of QCs do not have enough qubits to solve problems large enough to affect Bitcoin, different alternatives for the architecture of QCs are being considered, tested and implemented [19,53,54] so a sudden improvement in the approach might lead to a powerful quantum computer appearing virtually overnight.

As the above context is increasingly appreciated by members of the Bitcoin community, a number of informal discussions about ways to make Bitcoin adapt to a post-quantum world have recently been taking place online. While our methods were developed independently, the relevant discussions of which we have become aware are:

- Adam Back's mention of Johnson Lau's two-stage commitment method on Twitter [6].
- Tim Ruffing's scheme described in online conversations on the Bitcoin-dev Mailing list [48–50].
- Tristan Hoy's article on Medium [51,52], which arguably represents the closest proposal to ours.

In this paper we provide an overview of the potential impacts the emergence of quantum computers could have on Bitcoin. As such, we describe how a quantum-capable adversary is in the position of stealing funds from users who have revealed their public keys. Consequently, we propose a commit–delay–reveal protocol for the secure transition from Bitcoin's current signature scheme to a quantum-resistant signature scheme, applicable even if ECDSA has already been compromised. In contrast to existing proposals, we emphasize the necessity of a substantial delay phase to provide sufficient protection against accidental and, especially, adversarial chain reorganisation. We assume that the Bitcoin community has agreed on and deployed a quantum-resistant signature scheme, either as measure of precaution or as reaction to the appearance of a (fast) quantum-capable adversary. Independent of quantum computing, our protocol can be generally applied to react to the appearance of vulnerabilities rooted in Bitcoin's public key cryptography. The transition can be implemented as a soft fork using a similar approach as, for example, SegWit [27].

The remainder of this paper is organised as follows. Section 2 outlines the workings of Bitcoin and and provides an introduction on quantum computing. Section 3 examines the threats a quantum-capable attacker could pose for Bitcoin. In Section 4 we propose a protocol for the transition from Bitcoin's current signature scheme to a quantum-resistant one, while discussing the implementation details in Section 5. We conclude our paper in Section 6.

2 Background

In the following sections we provide relevant background on the workings of Bitcoin, its underlying cryptographic principles, as well as core quantum computing concepts, relevant for this paper. However, due to space limitations, we do not aspire to provide a complete description and hence recommend readers unfamiliar with these research fields to consult existing literature, such as [32, 33] for Bitcoin and [24, 36] for quantum computing.

2.1 Bitcoin and Blockchain

In Bitcoin, every transaction consists of inputs and outputs[3]. Each input references some unspent transaction output (UTXO) and provides a spending script (`scriptSig`) which will be used to authorize the transfer of funds. Each output is secured by a challenge script (`scriptPubKey`) which must be solved by the spending script of an input that wants to consume the funds. Based on the challenge script, one can distinguish different types of outputs[4]:

- **pay-to-pubkey** (`P2PK`) outputs were used before the concept of an address appeared. The challenge script contains the public key (p_k) associated with the secret key (s_k). An input wishing to consume such an output has to provide a digital signature of the transaction. If the signature can be verified against p_k, this means that it was indeed created by s_k, so the transfer of funds is valid.
- **pay-to-pubkeyhash** (`P2PKH`) outputs (presented in user interfaces as "addresses") are 160-bit hashes of the public keys [15]. This has the advantage of saving some space as addresses are shorter than public keys. To consume this type of output, an input needs to provide both the public key that hashes to the address and a digital signature that can be verified with the public key.
- **pay-to-scripthash** (`P2SH`) outputs (presented to the users as a new type of "address") are the hash of a script in which the user can specify different conditions to be satisfied by the input `scriptSig`. One use for this type of addresses is to achieve a compact UTXO format for multi-signature transactions.

Hence, a transaction takes some UTXOs as the source of funds and outputs new UTXOs, associated with the same or a different public key[5]. The concept of this transaction structure is fundamental in Bitcoin because it prohibits double spending of funds, as transaction outputs will be marked as spent and unspent, respectively, and hence no UTXO can be included by more than one transaction.

As part of Bitcoin's underlying consensus mechanism, termed Nakamoto consensus, a miner is asked to find the hash h of a block header which includes some

[3] With exception of the first transaction in each block in which new units of Bitcoin are released, termed coinbase transaction.

[4] There are actually many more types of challenge scripts and in Section 5 we briefly discuss how to make our transition protocol general enough to secure them all.

[5] Using a different public key is recommended for security and privacy reasons.

random input, or nonce, along with entities such as the hash of the previous confirmed block, such that h is below a difficulty threshold. The network difficulty is dynamically adjusted every 2,016 blocks such that the average block interval is approximately equal to 10 minutes. As finding a valid nonce is a memoryless process, the best known strategy for generating a PoW solution is the enumeration of all possible inputs and is therefore very computationally expensive. On the other hand, other nodes of the network can verify the proof-of-work criterion trivially with a single hash.

2.2 Elliptic Curve Digital Signature Algorithm (ECDSA)

ECDSA is an implementation of the Digital Signature Standard (DSS) based on Elliptic Curve Cryptography [5]. The purpose of such signatures is to allow third parties to determine the legitimacy and integrity of a signed message, while the signer cannot reasonably deny the act of signing. In Bitcoin, transactions are digitally signed using ECDSA, thus securing the transfer of ownership of bitcoins [11].

Elliptic Curve Cryptography (ECC) is a form of public-key cryptography that uses the mathematical properties of elliptic curves over finite fields [5]. More specifically, to define an elliptic curve cryptosystem one chooses a curve C and a public point P on the curve. To generate a pair of keys, one chooses a random number sk as the private key and uses elliptic curve point multiplication [5] to multiply the point P with itself sk times thus obtaining the public key pk which is itself another point on C. ECDSA or, in general, ECC, relies on the assumption that it is intractable to solve the elliptic curve discrete logarithm problem (ECDLP) [39], which would allow for deducing the private key from the public key. Like integer factorisation [18], ECDLP has no known reasonably fast (e.g. polynomial-time) solution on a classical computer [30].

2.3 Quantum Computing

Quantum computing makes use of various quantum phenomena such as superposition and entanglement to represent classical data in a quantum context and to manipulate it in ways that produce interpretable results [40]. Just like the state of classical computers is made of bits, quantum computers use qubits that have two fundamental (basis) states (0 and 1). However, while the computation is running, the state is a linear combination (superposition) of basis states, each having an associated probability to be measured.

To extract information about the state of a quantum computer (QC), the system is measured collapsing the superposition to one of the possible basis states. This means a QC with n qubits can represent internally the whole range of n-bit numbers and can perform calculations on all of them simultaneously; however, when measured, the state will collapse to just one of the basis states, thus returning only one of the results to the performed calculation. Therefore, instead, quantum algorithms try to make use of the underlying structure of the problem in order to amplify (or otherwise home in on) certain basis states, to increase their probability, and thus to make the result obtained repeatable and conclusive. For some problems, quantum algorithms can yield a significantly improved runtime complexity over their classical equivalents, thus offering a speed up.

Shor's Algorithm Shor's algorithm for integer factoring is a quantum algorithm with a runtime complexity of $O((\log N)^2(\log\log N)(\log\log\log N))$ [45], which is exponentially faster than all known classical algorithms. In fact, the integer factorisation problem can be reduced to finding the period of $f(x) = a^x \bmod N$ where a is a random integer and N is the number to be factored [26].

The algorithm works by preparing a superposition of basis states where each basis state is formed by concatenating x with the value of $f(x)$. When the qubits that store $f(x)$ are measured, the superposition will collapse leaving some value v on the qubits measured, while the qubits on which x was stored will be in a superposition of different x's with $f(x) = v$. To obtain the period of the function, the remaining algorithm needs to extract the difference between any of the states in the superposition. The quantum Fourier transform circuit can be used to achieve exactly this [26]. Shor's algorithm drastically weakens the security of some public-key cryptographic systems such as RSA, but Proos and Zalka show how it can be adapted to solve ECDLP with even fewer steps [38], offering a polynomial-time attack against ECDSA [45].

Grover's Algorithm Grover's algorithm is another efficient quantum computation. It aims to solve the problem of searching unstructured data by computing with high probability a unique (or very rare) solution x for which $f(x)$ equals v, some desired value [22]. The time complexity is $O\left(\sqrt{\frac{N}{t}}\right)$ where N is the size of the domain of f and t is the number of solutions [16]. The algorithm works by first arranging a superposition of all possible input states, each having equal probability of being measured. Then, it uses some techniques to iteratively increase the probability amplitude of the states that represent the solution [22]. Given N and t, the number of iterations after which the probability amplitudes of the correct states become maximal can be mathematically computed [16]. In case t is unknown there exists a scheme which will produce a solution in $O\left(\sqrt{\frac{N}{t}}\right)$ steps [16].

Note that it is not possible to measure the state after each iteration as this would collapse the superposition and the computation would end. Grover's algorithm is particularly interesting for mining as it theoretically offers a quadratic speed up when guessing a nonce. However, in practice, it is believed that early generations of QC will be slower than the optimised ASIC miners [7] [47].

2.4 Post-Quantum Cryptography

Post-quantum cryptography is a new branch of cryptography interested in a suite of algorithms which are believed to be secure even against attackers equipped with quantum computers [4]. There have been multiple proposals of cryptographic systems which are not yet broken by QC. Some examples are:

1. Code-based cryptography relies on the intractability of decoding unknown linear error-correcting codes [44]. McEliece used the algebraic properties of Goppa codes and proposed the first such system [28], which took his name.

2. Hash-based cryptography is based on the security of hash functions which, as mentioned, are not drastically weakened by QC. Merkle [31] was the first to propose hash-based digital signatures by building on the concept of one-time signature schemes such as Lamport's signature scheme [25].
3. Lattice-based cryptography is based on the hardness of lattice problems such as approximating the closest vector problem in a lattice [35].

For the purposes of our paper, it is important that the Bitcoin community agrees on and implements an appropriate alternative (or perhaps more than one) to replace Elliptic Curve Cryptography as the basis for digital signatures of transactions.

3 Bitcoin in a Post-Quantum World

Given that we assume a powerful QC could appear at any time, where does this leave Bitcoin? As presented in Section 2.3, in a post quantum world, miners could gain an unfair advantage by mining blocks using Grover's algorithm. This provides a quadratic speed-up in the number of operations compared to a classical computer, which should lead to an increased hashrate. However, current miners use parallel computations on optimised hardware (ASICs) and it is hence difficult to predict if and when quantum computers will be reliable and fast enough to outperform them. To this end, we assume that early generations of QCs will not be capable of outperforming classical miners in terms of hash rate. Furthermore, once QCs reach a state of development acceptable for mining, a quick adoption among miners can be expected, establishing an equilibrium as the network difficulty adjusts.

In this paper we do not aim at addressing potential vulnerabilities rooted in Bitcoin's PoW but rather at examining the risks quantum-capable attackers could pose for the embedded transaction processing mechanism. Once efficient quantum computers with internal states comprised of many qubits are implemented, the underlying cryptographic guarantees of Bitcoin can be challenged. As briefly mentioned in Section 2.3, an attacker with a quantum computer of about 1500 qubits [38, 47] can use Shor's algorithm to solve the ECDLP and compute an ECDSA private key given the public key, and is thus able to plant fake transactions and perform double-spending attacks. In the following sections we highlight why Bitcoin users should be concerned about exposing their public keys and describe a potential attack scenario whereby a quantum-capable attacker (QCA) engages in (live) transaction hijacking.

3.1 Public key unveiling

Under the assumption that quantum computers are being employed for malicious intent by some adversary, previously revealed public keys pose a direct threat to Bitcoin users. As outlined, a QCA is capable of deducing the private key from a former revealed public key with little effort. Such a scenario could arise from the following instances of public key unveiling:

1. Bitcoin transactions with `P2PK` UTXOs, as these display the public key in the output of the transaction. As soon as such a transaction has been included

in the blockchain, or even just broadcast to the network, a slow QCA can compute the corresponding private key and thereby essentially gain control over the respective funds. This problem can be mitigated by using, for example, `P2PKH` and `P2SH` addresses. However, when consuming such a UTXO, the owner of the address must reveal her public key and digital signature in the `scriptSig` of the respective input. Once this transaction is broadcast to the network for confirmation and inclusion in a block, the attacker can compute the private key from the revealed public key. Furthermore, the attacker could then look for any additional UTXOs associated to the same address and consequently consume them.
2. Bitcoin users publishing their public key on a Bitcoin fork, e.g. Bitcoin Cash [1] or Bitcoin Gold [2]. As Bitcoin forks share the same transaction history prior to the fork point, such behaviour may allow a QCA to gain control over a user's Bitcoin funds using the exposed public key. Furthermore, a QCA could then also exert control over funds on the blockchain where the public key was initially obtained, i.e. Bitcoin Cash.
3. Any other revealing of public keys, such as part of signed messages to ensure integrity, in forums, or in payment channels (e.g. Lightning Network [37]).

Regardless of how a public key is revealed, given the presence of a QCA, the owner is at risk of losing control over her funds. Except for `P2PK`, one can prevent against the aforementioned scenarios (so long as the QCA is slow to deduce a private key) by using addresses only once. Reusing addresses is not recommended, neither by Bitcoin developers nor the community, while numerous studies identifying privacy risks have been conducted [10,21,29,43,55]. Hence, we assume appropriate protective mechanisms are already employed by the majority of Bitcoin users.

3.2 Transaction hijacking

We assume that a fast QCA is characterised by the ability to perform (live) transaction hijacking. Thereby an attacker attempts to compute the private key corresponding to a public key revealed in the input of a transaction published to the network and sitting in nodes' memory pools. Consequently, just like in a double spending attack [8,23,41,46], she creates a conflicting transaction spending the same UTXOs[6] (or the subset the QCA has gained control over), thus stealing the victim's funds. As the attacker must not only create, sign and broadcast the conflicting transaction, but also first run Shor's algorithm to derive the private key, timing is essential for such attacks. Hence, the performance of QCs plays a central role for the success probability of transaction hijacking. Note that this form of transaction hijacking differs from the more conventional notion of double spending as the attacker is the sole beneficiary rather than the original transaction initiator.

While double spending may potentially only be economically feasible for high value transactions, an adversary with motives other than economic profit could

[6] Possibly with a higher fee to incentivise inclusion in the blockchain over the victim's transaction

use this to perform denial of service attacks and/or hinder the transition to a quantum-resistant signature scheme (as will be described in Section 4). Note we do not discuss the possibility of using Grover's algorithm to retrieve the public key from an address here, as the achieved speed-up is merely quadratic and can be mitigated by increasing the key size [7].

An extension to the described attacks is to combine transaction hijacking with selfish mining strategies [20, 21, 34, 42]. Assuming the QCA is also a miner, she could employ her computational power to attempt to build up her own secret chain and, when in the lead, selectively publish blocks to cause a reorganisation of the public chain. In contrast to traditional selfish mining attacks, the feasibility of such strategies under the presence of a QCA is expected to improve significantly, since the adversary can now also perform transaction hijacking as described above. The prospectively-gained revenue consists not only of block rewards and transaction fees, but also of all funds contained in (non-quantum-resistant) UTXOs spent in the overwritten transactions.

4 Transition to Quantum Resistance

In this section we describe a scheme which allows a secure transition from Bitcoin's current signature scheme to a quantum-resistant one. We assume a quantum-resistant signature scheme has already been agreed upon by the community, and deployed as a protocol update in Bitcoin. However, it is presumably unreasonable to hope that all Bitcoin users will have moved their coins from non-quantum-resistant to quantum-resistant outputs; inevitably, some people (perhaps even a majority) will still have control over non-quantum-resistant outputs, especially the most popular `P2PKH`. The protocol described in the following sections is designed to allow such users to transition securely, if rather slowly, to quantum-resistant outputs even in the presence of a fast QCA. It is based on a simple commit–delay–reveal mechanism with a long security delay, and can be deployed in Bitcoin using a soft fork. Bitcoin-specific implementation details, as well as discussion on parametrisation and necessary data structures are considered separately in Section 5. Note that if one uses an old client and spends from a non-quantum-resistant public key, the respective funds will be lost and no protective mechanism can be applied effectively.

4.1 Protocol Overview

Assume a user, Bob, is in possession of units of Bitcoin (BTC) stored in a non-quantum-resistant output, the public key of which has not yet been revealed, i.e., funded by an unspent `P2PKH` or `P2SH` output[7]. We shall denote Bob's public key as pk and the corresponding secret key as sk. Further, assume Bob has already generated a quantum-resistant keypair (pk_{QR}, sk_{QR}), which will be used to replace his current address as part of the transition. To convince the network he is the rightful controller of both keypairs and this way move funds to the quantum-resistant address, Bob publishes a commitment $H(pk|pk_{QR})$, i.e., the

[7] Our protocol actually caters for spending any number of such UTXOs in one transaction, but for simplicity we will consider here the spending of only one.

hash of his concatenated public keys, and leaves the funds on pk untouched for a sufficiently long security period t_{sec}. Once the period has passed, Bob creates a second transaction T_{reveal} signed by sk_{QR} which consumes the UTXOs attributed to (pk, sk) and reveals both public keys pk and pk_{QR}, proving to the network that he is the controller of both keypairs and signaling the transition of funds. We describe each step of this process in more detail in the following paragraphs.

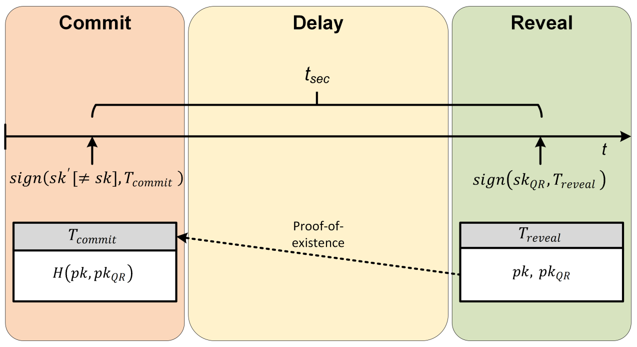

Fig. 1. Simplified visualization of the commit-delay-reveal transition scheme.

4.2 Commit

As a first step, to signal the commitment of the funds in (pk, sk), Bob publishes the hash of both public keys pk and pk_{QR} concatenated: $H(pk|pk_{QR})$. This is achieved by creating a transaction T_{commit}, which includes the hash commitment as an output. It is left for the user to decide upon the exact format of including the hash commitment in the blockchain. In Bitcoin, this can be achieved, for example, by using the `OP_RETURN` opcode, which allows to store up to 80 bytes of arbitrary data in a transaction [12].

The only secure way for Bob to perform such a T_{commit} would be to create a new address adhering to the quantum-resistant signature scheme with keypair (pk'_{QR}, sk'_{QR}) and acquire an arbitrary amount of quantum-resistant BTC, e.g., which were initially mined directly to another quantum-resistant address. Bob can then use sk'_{QR} to sign transactions, without risk of losing funds to a quantum-capable adversary.

4.3 Delay

After publishing the hash commitment, Bob leaves the funds in (pk, sk) untouched for a sufficiently long security period t_{sec}. Any further attempted use of this keypair, which would fail in accordance with the new protocol rules, puts Bob's funds at risk of theft. A long delay, is necessary to ensure no blockchain reorganization could have occurred accidentally or have been caused intentionally by an adversary. While the specific choice of delay may be subject to follow-up scientific work and discussion in the community, we propose an initial period of 6 months. A more detailed discussion is provided in Section 5.

4.4 Reveal

Once the security period has elapsed, Bob proceeds to reveal his public keys pk and pk_{QR}, proving to the network he is the rightful controller of both keypairs. To this end, Bob creates a transaction T_{reveal} signed by the secret key sk_{QR} of the new quantum-resistant keypair, which consumes the UTXOs of (pk, sk) and in which he

1. Reveals his "old" non-quantum-resistant public key pk,
2. Reveals the public key of the new quantum-resistant keypair pk_{QR},
3. Provides proof that he has published $H(pk|pk_{QR})$ in a transaction older than the security period t_{sec}.

Miners, adhering to the new protocol rules, will then be able to verify the funds that have been committed for a sufficient period to require a new quantum-resistant public key for their eventual spending. Hence, Bob will be allowed to spend his funds by providing a valid signature against his new quantum-resistant public key. Unupgraded consensus participants will simply believe T_{reveal} is a normal transaction consuming the UTXOs of (pk, sk). The necessary implementation specifics are provided in Section 5.3. As a result, the protocol update $P \rightarrow P'$ can be deployed as a soft fork, since the set of blocks valid under new rules P' is a proper subset of blocks valid under current Bitcoin rules P, i.e., $P' \subset P$.

5 Discussion

In this section we discuss selected implementation details of the introduced commit–delay–reveal transition scheme, including the choice of the delay period and the structure of the commit and reveal transactions.

5.1 Necessity for a Long Delay Phase

The correct choice of the security period t_{sec}, used as protection against accidental and adversarial chain reorganisations, has a significant impact on the security properties of the proposed transition protocol. In contrast to previous proposals and discussions [6,48–52] we emphasize the necessity of a sufficiently long delay phase, substantially longer than the standard confirmation period of $\sim$6 blocks in Bitcoin. While the exact duration of t_{sec} may be subject to future discussion, we propose to require hash commitments to be older than *6 months*, i.e., the UTXOs used as input to T_{reveal} must remain unspent during this period.

As explained in Section 3.2 we assume that the feasibility of block reorganisation attacks, such as 51% attacks or selfish mining attacks requiring a smaller fraction of the overall computational power, is significantly increased for quantum-capable adversaries. In contrast to traditional reorganisation attacks, the prospective gains in this scenario are not only comprised of block rewards and transaction fees but also include any funds associated with accounts whose public keys have been revealed in one of the blocks overridden by the attacker.

Hence, relying on a short security period of a few blocks (or no delay at all) provides insufficient protection against chain reorganisations in the presence of a quantum-capable attacker.

We note that in theory an adversary controlling a significant portion of the overall computational power could successfully rewind the chain further than t_{sec}, thereby altering the transaction history, and attempt to steal funds from all non-quantum-resistant outputs which were spent from during this period. However, we argue a fork overriding the block history of such substantial period as 6 months would be classified as a catastrophic failure of the system, forcing out-of-band measures to be undertaken by the majority of honest consensus participants. Specifically, we assume clients and miners will have incentive to manually reject the conflicting branch of the attacker[8].

However, by intuitive continuity arguments there must exist a point between short- and long-ranged attacks, where the community is unable to find even out-of-band consensus on how to proceed, i.e., whether to perform a manual invalidation (override of attacker's fork) soft fork or accept the conflicting branch of the adversary, as visualized in Figure 2. While under different circumstances, similar disputes have been observed in other cryptocurrencies and have led to permanent chain splits, as in the case of Ethereum [17] and Ethereum Classic [3]. Hence, a quantum-capable adversary may have incentive to attempt to exploit this "sweet-spot" to her advantage, as a destabilization or split of the chain could yield a higher success probability of an attack.

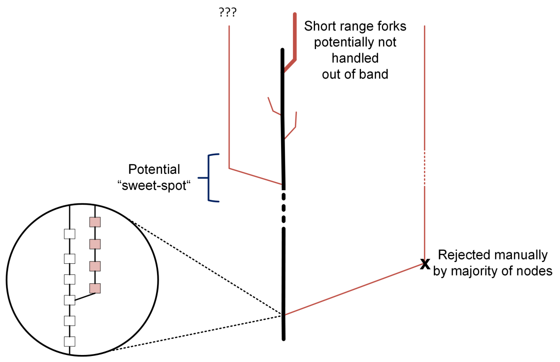
???
Short range forks potentially not handled out of band
Potential "sweet-spot"
Rejected manually by majority of nodes

Fig. 2. While long-range forks are expected to be manually rejected by the majority of nodes, this may not be possible with short-range chain-splits due to the limited time frame. There may exist a "sweet-spot" which causes a dispute whether to accept or reject the conflicting branch, destabilizing or even permanently splitting the network to the benefit of the adversary (red).

[8] Note that this does not require any changes to the reference client implementation, as Bitcoin's JSON-RPC API provides a `invalidateblock` call, which permanently marks a specific block as invalid, as if it had violated a consensus rule [13].

By implementing a long delay phase, sufficient to trigger out-of-band actions in case a longer fork is created by an adversary, the probability of a malicious chain reorganisation interfering with the transition protocol can be minimized.

Arguments Against Parametrisation by Users Instead of defining the delay phase t_{sec} as part of the consensus rules, a naïve approach would be to allow each user to declare their own security period. However, we emphasize the necessity of a global fixed delay phase, as allowing individual parametrisation by users leaves the transition scheme vulnerable to attacks, as described in the following:

Declare t_{sec} during Reveal: A straightforward approach would be for users to declare their own security period t_{sec} during the reveal phase, i.e., in T_{reveal}. The problem with this approach, however, is that a quantum-capable adversary can then derive the user's secret key sk from the now revealed public key pk. The attacker will then attempt to revert the public chain so that T_{reveal} is no longer included in the blockchain, and create a new hash commitment $H(pk, pk'_{QR})$ for her own address pk'_{QR} by publishing T'_{commit}. After waiting for a minimal period to be sure the majority of consensus participants has accepted the attacker's chain, she moves on to reveal the victim's (non-quantum-resistant) and her own (quantum-resistant) public key in T'_{reveal}, declaring a very short security period. As a result, the victim's funds in (pk, sk) will be transferred to the attacker's address. Note that it is sufficient for the adversary to be able to revert only a few blocks for this attack to be successful.

Declare t_{sec} during Commit: A possible way of mitigating the attack described above is to require users to declare t_{sec} as part of the hash commitment and employ a "first-seen" rule, i.e., only consider the first commitment included in the blockchain for each address as valid. However, in order for clients to be able to link the reveal transaction to the correct address and verify the individually set delay phase when finalizing the transition, users are required to make their hash commitments publicly verifiable, i.e., include the hash of their public key in T_{commit}[9]. While this approach prevents an attacker from overriding a user's reveal transaction, it also enables *griefing*. As such, an adversary could easily publish fake commitments containing arbitrary data for the supposed hash of the pairing for any known addresses in Bitcoin, preventing the transition of funds altogether.

5.2 Structure of the Hash Commitment

During the commit phase of the protocol, a transaction T_{commit} containing the hash commitment for pk and pk_{QR} is created. As mentioned, the Bitcoin `OP_RETURN` script operation allows to push up to 80 bytes of arbitrary data onto the stack [12], which is sufficient to, for example, persist a SHA-256 hash. However, there may also be alternatives to this way of publishing the hash commitment.

[9] Signing the transaction with pk would reveal the public key, instantly making the funds vulnerable to theft by a quantum-capable adversary

The exact format of the hash commitment, having little impact on the introduced transition protocol, is expected to be subject to an open discussion in the community. For simplification, we propose to use the concatenation of the public keys pk and pk_{QR} as input to the hash function. This, however, assumes that agreement on the quantum-resistant signature scheme has been reached beforehand.

5.3 Reveal Structure and Backward Compatibility

During the reveal phase of the transition protocol, users must prove to the network they are the rightful controllers of the non-quantum-resistant keypair (pk, sk) and the quantum-resistant keypair (pk_{QR}, sk_{QR}), and provide evidence that there exists a hash commitment for the public keys of these accounts older then the security period t_{sec}. The latter is achieved by providing a SPV (Simplified Payment Verification) proof [9,14], i.e., including the path to T_{commit} in the Merkle Tree transaction structure of the respective block, dating back t_{sec} or more, in the reveal transaction T_{reveal}.

To enable the deployment of the transition scheme as a soft fork, i.e., without requiring a permanent split of the blockchain, we propose a scheme similar to that used in SegWit [27]. As such, the data witnessing the new rules are being obeyed is held in a segregated area, termed *QRWitness*, which new clients receive and check but old clients remain oblivious to. To make sure the witness structure is part of the hash of the block it is contained in, the root of a Merkle Tree consisting of all QRWitness-es is inserted in the respective coinbase transaction. While the original transaction `txid` remains the same as before, a new `qrtxid` is defined as the double SHA256 hash over the traditional transaction format *and* the QRWitness. Thereby, a possible format for QRWitness could be the following:

```
<oldPubkey><pubkeyQR><merklepath><signatureQR>
```

where `oldPubkey` denotes the non-quantum-resistant public key pk, `pubkeyQR` is the quantum-resistant public key pk_{QR}, `merklepath` represents the path to the hash of the T_{commit} transaction and `signatureQR` denotes the signature of the traditional transaction format using sk_{QR}.

To achieve backward compatibility, the `scriptSig` field remains such that it satisfies the consensus rules of old clients, e.g., the non-quantum-resistant signature and the corresponding public key. This way, just like SegWit, our transition protocol can be deployed as a soft fork in Bitcoin. Note however, that as usual with soft fork attempts, if the majority of the mining power does not upgrade and continues to accept transactions spending non-quantum-resistant outputs without adhering to the commit–delay–reveal structure, the soft fork will cause a potentially permanent split of the blockchain.

The extension to more diverse challenge scripts than are covered by the usual address types should now be clear; namely, provide in the QRWitness as many instances of `<oldPubkey><pubkeyQR><merklepath><signatureQR>` as are necessary, i.e., one for each act of checking a non-quantum-resistant signature under the old rules.

6 Conclusion

In light of the emerging threat of quantum-capable adversaries in Bitcoin, we have outlined how Bitcoin could become subject to theft of funds rooted in the exposure of public keys. Thus, we have proposed a commit–delay–reveal scheme to allow for the secure transition to a quantum-resistant address scheme in Bitcoin, the underlying protocol modifications for which can be implemented as a soft fork. For the security of the transition scheme we emphasize the need for a sufficiently long delay period and propose an initial period of 6 months in order to prevent possible blockchain reorganisation. The proposed time frame should suffice for allowing honest clients and miners to reach consensus on manually rejecting long range forks that exceed the delay period. However, we suggest that by intuitive continuity arguments there must exist some point in time where the community would be indecisive on how to proceed given that a conflicting branch created by an adversary exists. Hence, we note that the optimal duration of the delay period may be subject to future discussion and analysis.

7 Acknowledgements

The authors would like to thank João Ribeiro and Dimitrios Myrisiotis for helpful comments and insightful discussions.

References

1. Bitcoin Cash. `https://www.bitcoincash.org/`. Accessed: 2018-02-18.
2. Bitcoin Gold. `https://bitcoingold.org/`. Accessed: 2018-02-18.
3. Ethereum Classic. `https://ethereumclassic.github.io/`. Accessed: 2018-02-18.
4. Introduction to post-quantum cryptography. `http://www.pqcrypto.org/www.springer.com/cda/content/document/cda_downloaddocument/9783540887010-c1.pdf`. Accessed: 2018-02-20.
5. X9-Financial Services Public Key Cryptography For The Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA) ©. 1998.
6. Adam Back. `https://twitter.com/adam3us/status/947900422697742337`. Accessed: 2018-02-18.
7. M. Amy, O. Di Matteo, V. Gheorghiu, M. Mosca, A. Parent, and J. Schanck. Estimating the cost of generic quantum pre-image attacks on sha-2 and sha-3. In *International Conference on Selected Areas in Cryptography*, pages 317–337. Springer, 2016.
8. E. Androulaki, S. Capkun, and G. O. Karame. Two bitcoins at the price of one? double-spending attacks on fast payments in bitcoin. In *CCS*, 2012.
9. A. Back, M. Corallo, L. Dashjr, M. Friedenbach, G. Maxwell, A. Miller, A. Poelstra, J. Timón, and P. Wuille. Enabling blockchain innovations with pegged sidechains. `http://newspaper23.com/ripped/2014/11/http-_____-___-_www___-blockstream___-com__-_sidechains.pdf`, 2014. Accessed: 2016-07-05.
10. A. Biryukov, D. Khovratovich, and I. Pustogarov. Deanonymisation of clients in bitcoin p2p network. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 15–29. ACM, 2014.

11. Bitcoin community. Elliptic Curve Digital Signature Algorithm. `https://en.bitcoin.it/wiki/Elliptic_Curve_Digital_Signature_Algorithm`. Accessed: 2018-02-18.
12. Bitcoin community. OP_RETURN. `https://en.bitcoin.it/wiki/OP\_RETURN`. Accessed: 2018-02-18.
13. Bitcoin community. Original Bitcoin client/API calls list. `https://en.bitcoin.it/wiki/Original_Bitcoin_client/API_calls_list`. Accessed: 2018-02-18.
14. Bitcoin Community. Simplified payment verification. `https://bitcoin.org/en/developer-guide#simplified-payment-verification-spv`. Accessed: 2018-02-18.
15. Bitcoin Community. Technical background of version 1 bitcoin addresses. `https://en.bitcoin.it/wiki/Technical_background_of_version_1_Bitcoin_addresses`. Accessed: 2018-02-18.
16. M. Boyer, P. Hoyer, and A. Tapp. Tight bounds on quantum searching. 1996.
17. V. Buterin. Ethereum: A next-generation smart contract and decentralized application platform. `https://github.com/ethereum/wiki/wiki/White-Paper`, 2014. Accessed: 2016-08-22.
18. R. E. Crandall and C. Pomerance. *Prime numbers: A computational perspective, Chapter 5*. Springer, 2005.
19. S. Debnath, N. M. Linke, C. Figgatt, K. A. Landsman, K. Wright, and C. Monroe. Demonstration of a small programmable quantum computer with atomic qubits. *Nature*, 536(7614):63, 2016.
20. I. Eyal and E. G. Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *Financial Cryptography and Data Security*, pages 436–454. Springer, 2014.
21. A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun. On the security and performance of proof of work blockchains. `https://eprint.iacr.org/2016/555.pdf`, 2016. Accessed: 2016-08-10.
22. L. K. Grover. A fast quantum mechanical algorithm for database search. In *Proc. 28th ACM Symposium on Theory of Computing (STOC '96)*, 1996.
23. G. O. Karame, E. Androulaki, M. Roeschlin, A. Gervais, and S. Čapkun. Misbehavior in bitcoin: A study of double-spending and accountability. *ACM Transactions on Information and System Security (TISSEC)*, 18(1):2, 2015.
24. P. Kaye, R. Laflamme, and M. Mosca. *An introduction to quantum computing*. Oxford University Press, 2007.
25. L. Lamport. Constructing Digital Signatures from a One Way Function. 1979.
26. C. Lavor, L. R. U. Manssur, and R. Portugal. Shor's algorithm for factoring large integers. `https://arxiv.org/pdf/quant-ph/0303175.pdf`, 2003. Accessed: 2018-02-07.
27. E. Lombrozo, J. Lau, and P. Wuille. BIP141: Segregated Witness (consensus layer). `https://github.com/bitcoin/bips/blob/master/bip-0141.mediawiki`, 2012. Accessed: 2018-02-18.
28. R. J. McEliece. A public-key cryptosystem based on algebraic coding theory. *The Deep Space Network Progress Report*, 42-44:114–116, 1978.
29. S. Meiklejohn, M. Pomarole, G. Jordan, K. Levchenko, D. McCoy, G. M. Voelker, and S. Savage. A fistful of bitcoins: characterizing payments among men with no names. In *Proc. 2013 Internet Measurement Conference*, pages 127–140. ACM, 2013.
30. A. Menezes. Evaluation of Security Level of Cryptography: The Elliptic Curve Discrete Logarithm Problem (ECDLP). 2001.

31. R. C. Merkle. A certified digital signature. In G. Brassard, editor, *Advances in Cryptology — CRYPTO' 89 Proceedings*, pages 218–238, New York, NY, 1990. Springer New York.
32. S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. `https://bitcoin.org/bitcoin.pdf`, Dec 2008. Accessed: 2015-07-01.
33. Narayanan, Arvind and Bonneau, Joseph and Felten, Edward and Miller, Andrew and Goldfeder, Steven. Bitcoin and cryptocurrency technologies. `https://d28rh4a8wq0iu5.cloudfront.net/bitcointech/readings/princeton_bitcoin_book.pdf?a=1`, 2016. Accessed: 2016-03-29.
34. K. Nayak, S. Kumar, A. Miller, and E. Shi. Stubborn mining: Generalizing selfish mining and combining with an eclipse attack. In *1st IEEE European Symposium on Security and Privacy, 2016*. IEEE, 2016.
35. P. Q. Nguyen and O. Regev. Learning a parallelepiped: Cryptanalysis of GGH and NTRU signatures. *Journal of Cryptology*, 2009.
36. M. A. Nielsen and I. Chuang. Quantum computation and quantum information, 2002.
37. J. Poon and T. Dryja. The bitcoin lightning network. `https://lightning.network/lightning-network-paper.pdf`, 2016. Accessed: 2016-07-07.
38. J. Proos and C. Zalka. Shor's discrete logarithm quantum algorithm for elliptic curves. `https://arxiv.org/pdf/quant-ph/0301141v2`, 2004. Accessed: 2018-02-07.
39. C. Research. Certicom ECC Challenge. `https://www.certicom.com/content/dam/certicom/images/pdfs/challenge-2009.pdf`. Accessed: 2018-02-17.
40. E. Rieffel and W. Polak. An introduction to quantum computing for non-physicists. *ACM Computing Surveys*, 32, 2000.
41. M. Rosenfeld. Analysis of hashrate-based double spending. `http://arxiv.org/abs/1402.2009`, 2014. Accessed: 2016-03-09.
42. A. Sapirshtein, Y. Sompolinsky, and A. Zohar. Optimal selfish mining strategies in bitcoin. `http://arxiv.org/pdf/1507.06183.pdf`, 2015. Accessed: 2016-08-22.
43. N. Schneider. Recovering bitcoin private keys using weak signatures from the blockchain. `http://www.nilsschneider.net/2013/01/28/recovering-bitcoin-private-keys.html`, 2013. Accessed: 2018-02-18.
44. N. Sendrier. *Code-Based Cryptography*, pages 215–216. Springer US, Boston, MA, 2011.
45. P. W. Shor. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *AT&T Research*, 1994.
46. Y. Sompolinsky and A. Zohar. Bitcoin's security model revisited. `http://arxiv.org/pdf/1605.09193`, 2016. Accessed: 2016-07-04.
47. L. Tessler and T. Byrnes. Bitcoin and quantum computing. `https://arxiv.org/pdf/1711.04235v2`, 2018. Accessed: 2018-02-08.
48. Tim Ruffing. `https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2018-January/015619.html`. Accessed: 2018-02-18.
49. Tim Ruffing. `https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2018-January/015659.html`. Accessed: 2018-02-18.
50. Tim Ruffing. `https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2018-February/015758.html`. Accessed: 2018-02-18.
51. Tristan Hoy. `https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2018-February/015715.html`. Accessed: 2018-02-18.
52. Tristan Hoy. `https://medium.com/@tristanhoy/11271f430c41`. Accessed: 2018-02-18.

53. M. Veldhorst, C. Yang, J. Hwang, W. Huang, J. Dehollain, J. Muhonen, S. Simmons, A. Laucht, F. Hudson, K. Itoh, et al. A two-qubit logic gate in silicon. *Nature*, 526(7573):410, 2015.
54. T. Watson, S. Philips, E. Kawakami, D. Ward, P. Scarlino, M. Veldhorst, D. Savage, M. Lagally, M. Friesen, S. Coppersmith, et al. A programmable two-qubit quantum processor in silicon. *Nature*, 2018.
55. Y. Yarom and N. Benger. Recovering OpenSSL ECDSA nonces using the FLUSH+RELOAD cache side-channel attack. *IACR Cryptology ePrint Archive*, 2014:140, 2014.