Blockchained Post-Quantum Signatures

Konstantinos Chalkias*, James Brown†, Mike Hearn‡, Tommy Lillehagen§, Igor Nitto¶, Thomas Schroeter‖

R3

Email: *konstantinos.chalkias@r3.com, †james.brown@r3.com, ‡mike@r3.com, §tommy.lillehagen@r3.com, ¶igor.nitto@r3.com, ‖thomas.schroeter@r3.com

***Abstract*—Inspired by the blockchain architecture and existing Merkle tree based signature schemes, we propose BPQS, an extensible post-quantum (PQ) resistant digital signature scheme best suited to blockchain and distributed ledger technologies (DLTs). One of the unique characteristics of the protocol is that it can take advantage of application-specific chain/graph structures in order to decrease key generation, signing and verification costs as well as signature size. Compared to recent improvements in the field, BPQS outperforms existing hash-based algorithms when a key is reused for reasonable numbers of signatures, while it supports a fallback mechanism to allow for a practically unlimited number of signatures if required. We provide an open source implementation of the scheme and benchmark it.**

***Index Terms*—post-quantum cryptography, digital signature, distributed ledger, blockchain, Merkle tree**

I. Introduction

Recent advances in quantum computing and the threat this poses to classical cryptography has increased the interest in PQ research. More specifically, due to Shor's algorithm [1], a quantum computer could easily factor a big integer in polynomial time, thus effectively break RSA. Implementations of Shor's algorithm can also solve discrete logarithms and render today's digital signatures, such as DSA, ECDSA and EdDSA, useless [2].

The race to build quantum computers has already begun and companies like Google, Microsoft, IBM, D-Wave and Intel are at the forefront. That being said, we have yet to build a computer with the thousands of stable qubits that are required to make classical public key cryptography obsolete. However, there is significant progress in the field and some optimistic predictions estimate that a large quantum computer capable of breaking asymmetric cryptography might be available in the next 10 to 20 years [3], [4].

The security impact of breaking public key cryptography would be tremendous, as almost everything from HTTPS, VPN and PKI in general, is basing their authentication, key exchange and digital signatures on the security of RSA or Elliptic Curve Cryptography (ECC). Blockchains would be hit equally hard resulting in broken keys that hold coins/assets, and would perhaps be one of the most affected sectors because there is economic incentive for hackers to get access to blockchain accounts anonymously.

To address the concern of compromised keys, PQ cryptography is dealing with the design and evaluation of systems that will survive the quantum supremacy. Our proposed BPQS solution is a modified version of the hash-based XMSS [5] family of schemes. It practically makes use of a single authentication path; thus, it is a chain and not a tree and it mainly focuses on {one and limited}-time keys, which is usually the most applicable to blockchains as we want to preserve anonymity and minimise tracking.

Compared to existing schemes, our approach outperforms limited-time schemes when required to sign only once or a few times. Unlike one-time schemes (OTS), BPQS schemes provide a fallback mechanism to easily support many-time signatures. Moreover, the underlying logic of a "blockchained" authentication path could be applied to convert any existing hash-based scheme to a {one and/or few}-time optimised one. To our knowledge, this is the first signature scheme that can utilise an existing blockchain or graph structure to reduce the signature cost to one OTS, even when we plan to sign many times. This makes existing many-time stateful signature schemes obsolete for blockchain applications. Moreover, the fact that BPQS is solely based on hash functions and that no special math theory is required for its implementation makes it a promising candidate for existing or new blockchain applications, and for low-end devices, such as in IoT applications, where hashing operations are already implemented and sometimes hardware-optimised.

II. Quantum Computing

Driven by market requirements for more time-efficient computing and the ability to solve problems that were previously considered impracticable, quantum computing is quickly moving from fundamental theoretical research to reality. 10 years ago there was little evidence of practical quantum computers. However, in 2018, Google unveiled Bristlecone [6], a new quantum computing chip with 72 qubits, 22 more than the 50-qubit processor announced by IBM in 2017. We should also mention D-Wave, a company that recently announced a 2000-qubit processor optimised for quantum annealing metaheuristics [7]; however, there are reports that D-Wave's quantum speedup analysis is debatable [8]. To summarise, even though more research and experiments are required to allow for stable quantum chips with low error rates, it is a fact that quantum technology is progressing.

A. Impact on Cryptography

Quantum technology has introduced new security challenges, and as mentioned above, raised the prospect of weakening classical cryptography. Due to Shor's algorithm, the widely used public key cryptography building blocks based on factorisation and discrete logarithms are considered broken in the PQ era. According to some estimations [3], there is a 50 % probability of breaking RSA and ECC by 2031. Furthermore, Grover's algorithm [9] might also affect symmetric encryption and hashing, but we currently do not know how to get more than a quadratic speedup over a classical computer, and thus they can be made PQ safe by just increasing the key/output sizes. We should also highlight that contrary to the quantum hype, there are sceptics [10] who believe quantum supremacy will never reach the level of thousands of usable logical qubits [2] that is required to break classical cryptography.

Despite the uncertainty of when and if large-scale quantum computers will become available, there is still significant research and development undertaken in parallel by both academia and the industry. There are also attempts at combining classical and PQ algorithms [11], [12] so that we are better prepared for the quantum apocalypse, if it ever happens.

It should also be mentioned that standardisation institutes, such as NIST, have started looking at standardising PQ algorithms [13], [14]. The European Telecommunications Standards Institute has been even more cautious and advised that any organisation with a need to archive encrypted data until 2025 or beyond should be worried about the threat of quantum computers [15]. The above raises concerns on the security-level advertised by blockchain and DLT solutions, mainly because public keys might hold assets/coins for decades.

B. Implications on Blockchains and DLTs

Conventional blockchains, such as Bitcoin and Ethereum, employ classical public key cryptography to sign transactions in their networks and they are considered vulnerable to quantum attacks. Other systems, such as zCash and Quorum, heavily rely on special elliptic curves to provide Zero Knowledge Proof functionality and an ECC breach threatens the integrity of their ledger [16].

Whilst this would be a significant vulnerability resulting in a total compromise of the network, most blockchains mitigate the threat by using addresses that are generated from cryptographic hashes of the public keys. This additional layer of security means that a public key only gets exposed to the ledger after the first transaction that it participates in occurs (*i.e.*, a Bitcoin is spent). Until this point, only the hashed recipient key (the address) is exposed and consequently attacks such as Shor's algorithm are not applicable at this stage.

However, the following attack vectors are still applicable, even when hashed keys are utilised:

- **Address reuse** – When a transaction is signed, the public key gets revealed and thus, the associated address is no longer safe. Despite the recommendation of using new addresses/keys for every transaction, older Bitcoin clients and some mining pools still reuse addresses.
- **Abandoned coins/assets** – If their associated addresses were generated without hashing, the public keys for these older addresses would become exposed, *e.g.*, prior to 2012 for Bitcoin.
- **In-flight transactions** – Once broadcast to the network and still waiting to be placed on the blockchain, transactions are vulnerable to attacks. Granted, the window of opportunity is limited, but it is still theoretically possible for an attacker to recover a private key and then sign another transaction that transfers the assets to their own address before the legitimate transaction is executed.
- **Transaction rejections/failures** – If a signed transaction does not go through, *e.g.*, due to low fees, a malicious party preventing a transaction from relaying, or a script failing during verification, the key will be left vulnerable to attacks.
- **Multi-sig transactions / transaction mixing** – As with the CoinJoin protocol [17], this will reveal the public keys to other parties before a transaction is finalised.
- **Advertisement of a single address** – Advertising and using the same address for, *e.g.*, fund raising will expose the public key on the first consuming transaction and therefore put all subsequent fund receipts at risk.

The commonly proposed solution is to upgrade the signing algorithms to be quantum-safe, however this will invariably result in a hard-fork of the blockchain which introduces various complexities. There are however blockchain solutions that already support post-quantum techniques, such as the Quantum Resistant Ledger (QRL) [18], IOTA [20] and Corda [22] in which BPQS could be applied to reduce signature sizes, while providing a fallback mechanism to allow signing with the same key multiple times.

III. Scenarios for Signing with the Same Key

OTS hashing schemes require that a key is used only once otherwise security becomes compromised. However, in a blockchain, there are practical scenarios that require multiple signatures using the same key:

- **Transaction failures and rejections** can occur as described in the previous section, and require additional signatures to resolve.
- **Proof of key ownership**, where a Party A is required to sign a message provided by a challenging Party B in order for Party B to validate the ownership.
- **Proof of solvency**, where proof of ownership of a minimum amount of an asset is desired, without leaking any further information. A solution to this is for an independent audit to be performed with results recorded in a "send-to-self" transaction signed with the owner's private key(s).
- **Forked blockchains**, where addresses (and associated keys) are replicated across forks and subsequently spent in each one of them – the duplicate transactions from

the same address (signed with OTS schemes) violate the condition that a key can be used only once and the strength of the OTS signature will halve as a result.

- **Strategic double-spending** of an asset, causing transactions to be rejected – whilst essentially a race condition, this scenario can be instigated to deliberately cause a pending transaction to fail. This might occur in the case of accidental spending for example. Issuing a duplicate transaction signed with the same key can cause both transactions to fail but has the undesired consequence that the key is re-used.

IV. Hash-Based Post-Quantum Digital Signatures

A. *One-Time Signatures*

Hash-based signature schemes have been documented in the literature since 1979, thanks to the Lamport OTS scheme [24]. The logic behind Lamport's scheme is straightforward; the signer generates pairs of random values per bit required to be signed and these pairs form the private key. The public key is formed by the hashes of those values. To sign a message, the signer reads the message bitwise and presents one value from each secret pair depending on the bit value. The verifier can then validate that the hashes of all the secret values are equal to the corresponding hash values in the public key.

Although Lamport OTS hash computations are considered fast, key and signature sizes are relatively large. For instance, if SHA256 is used as the underlying hash function, the public key consists of 512 hashed outputs of 256 bits each (one hash-pair per bit), while the signature consists of 256 secret values (256 bits each). If we aggregate the above, the key and signature consist of 24.5 kB. Similarly, if SHA512 is applied, about 98 kB are required.

Further enhancements to the original algorithm [19], [25], [26], reduce the key size significantly. At present, the WOTS [19] algorithm and its variants are considered some of the most efficient key and signature compression methods, while Bleichenbacher and Maurer's graph-based scheme [26] attempts to achieve the best possible efficiency in terms of signature size *and* number of hash function evaluations per bit of the message.

As a note, one of the main differences between OTS approaches lies in the security assumptions of requiring (or not) collision resistant hash functions and the use of extra bitmasks. Currently, WOTS-T [27], proven to be secure in the QROM model, is considered one the the most promising candidates from the WOTS [19] family, because only one extra seed value is required along with the public key to compute the required bitmasks, while its security is not affected by the birthday paradox and it also introduces keying of all hash function calls to prevent multi-target second pre-image attacks. The latter results in shorter public keys and hash-output sizes.

B. *Few- and Many-Time Signatures*

Although there exist multiple methods to turn a one-time into a multi-time signature scheme [5], [28]–[30], a popular approach is to use Merkle authentication trees by fixing beforehand the total number of signatures which will ever be produced. Using Merkle trees, the total number of signatures which can be issued is defined at key generation. The main benefit is its short signature output and fast verification, while the drawbacks are the relatively expensive key generation time and the fact that they are stateful. Figure 1 depicts a 4-time (at maximum) Merkle tree signature scheme.

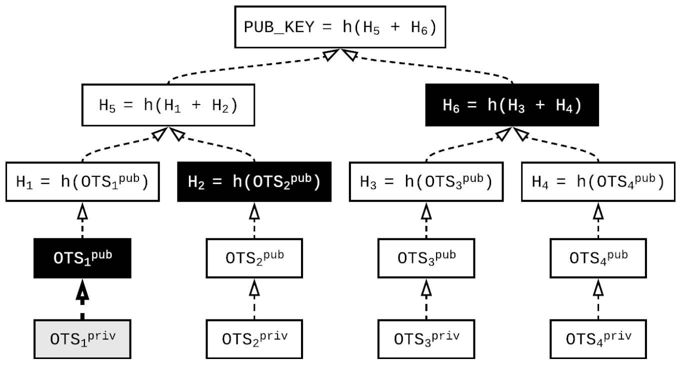

Fig. 1: Few-time Merkle tree signature scheme able to sign four messages in total. Dark nodes represent the authentication path required if we sign with OTS_1.

Moving to stateless few-time signatures requires extra complexity and larger signature outputs. HORS [29] (and its extension HORST [23]) is currently the one used in the majority of many-time stateless signature schemes, such as SPHINCS [23].

Many-time hash-based schemes can be constructed by combining the above {one and few}-time constructions and they are grouped into two categories, stateful (*e.g.*, XMSS, LMS) [31] and stateless (*e.g.*, SPHINCS, SPHINCS$^+$, Gravity, Simpira, Haraka) [23], [32]–[35]. Stateful schemes typically produce shorter signatures, but they need a mechanism to keep state (what paths/keys have already been used).

On the other hand, stateless schemes start with a moderately large Merkle tree or tree-layers at the top, but instead of using OTS signatures at the bottom, they use a few-time signature method. The latter allows them to pick indices randomly and thus no path-state tracking is required. The downside to stateless schemes is their signature size; for instance, in SPHINCS-256 [23] each signature is 41 kB long.

It is highlighted that the distinction between few- and many-time hash-based signature schemes is not always clear. In the literature, few-time usually refers to stateless schemes, such as BiBa [28], HORS [29] and HORST [23], for which practical parameters allow multiple signing operations, but not enough signatures to be considered in many real-world applications. On the other hand, many-time schemes can be configured to allow highly interactive environments to reuse the same key-pair for many years. The authors of Gravity SPHINCS [33] claim that 1 trillion (2^{40}) signatures is a reasonable upper bound, whilst SPHINCS-256 [23] allows for a maximum of 1 quadrillion (2^{50}) signatures. In practice, one can parameterise a many-time scheme to support just a few or several signatures.

C. Speed and Security of Hash Functions

The underlying hash algorithm is of obvious importance to the overall security of the proposed scheme. Several factors influence the choice of algorithm, including speed, security level and availability; *e.g.*, what hardware features can be leveraged to improve the runtime performance, and what implementations are available in existing, well-reviewed cryptography libraries.

The first thing to establish, however, is whether the algorithm is resilient to PQ attacks. The SHA-2 and SHA-3 algorithms support multiple digest sizes, namely 224, 256, 384 and 512 bits [36], [37]. We observe that by leveraging the improved search speed provided by Grover's algorithm, collision resistance can be reduced from a half to a third of the chosen digest size. Consequently, in the presence of large-scale quantum computers, 384-bit versions of SHA-2 and SHA-3 would provide 128 bits of security against collisions, whereas the 256-bit versions would only offer 85 bits [4].

Further, we observe that quantum pre-image attacks on 256-bit versions of SHA-2 and SHA-3 can be realised by $2^{153.8}$ and $2^{146.5}$ surface code cycles, respectively [38].

As a result of these two observations, SHA256 is considered unsuitable for use in schemes basing their security on hash collision resistance, but it is still secure otherwise. It should also be mentioned that PQ algorithms have fundamentally worse price-performance ratio than the classical van Oorschot-Wiener hash-collision circuits, even under optimistic assumptions regarding the speed of quantum computers [39].

From performance measurements presented in eBACS [40], we have evaluated the relative performance of SHA-2, SHA-3 and BLAKE2 on general-purpose CPUs. We have deliberately chosen an Intel, an AMD and an ARM processor to cover typical desktop and mobile units.

As can be seen from Table 1, the number of cycles per byte decreases with the size of the input. This is expected due to the small input sizes in this comparison and the block-wise operation mode of the hash functions. The rate of decrease naturally flattens out as the input grows beyond the block size.

It should also be noted that the different versions of SHA-3 generally performs worse than their SHA-2 counterparts. One of the reasons for this is the fact that SHA-1 and SHA-2 have better hardware support from modern processors, *e.g.*, through instruction set extensions like the Intel® SHA Extensions.

Note that, despite not offering protection against length extension attacks, SHA-2 offers similar bit-level security to SHA-3. Typically, hash-based PQ schemes, including BPQS, are not prone to such attacks and therefore, we consider SHA-2 to be a better alternative due to the performance benefits it offers.

If performance is of importance, one can also consider employing the less supported BLAKE2b [41] algorithm. We highlight, however, the lack of wide-spread library support compared to the aforementioned algorithms.

Table 1: Performance Metrics for SHA-2, SHA-3 and Blake

Input Size	**Measurements of Hash Functions**[a] Cycles / Byte (relative to SHA2-256 on 8 byte input)								
	SHA-2			**SHA-3**			**BLAKE2**[b]		
	Intel	AMD	ARM	Intel	AMD	ARM[c]	Intel	AMD	ARM
256-bit Output									
8	1.00	0.19	2.99	3.48	2.89	6.78	0.47	0.38	2.30
64	0.24	0.04	0.54	0.46	0.38	0.85	0.05	0.04	0.28
576	0.13	0.02	0.20	0.25	0.21	0.40	0.05	0.04	0.15
512-bit Output									
8	1.49	1.12	5.72	3.58	3.00	6.79	0.53	0.46	3.23
64	0.19	0.14	0.71	0.48	0.40	0.85	0.07	0.05	0.41
576	0.09	0.05	0.30	0.43	0.36	0.71	0.03	0.03	0.14

[a]Based on numbers reported by ECRYPT II in eBACS [40].
- **Intel** - amd64, genji122, supercop-20171020
- **AMD** - amd64, genji262, supercop-20171020
- **ARM** - armeabi, odroid, supercop-20160806

[b]BLAKE2s with 32-bit words, 10 rounds, and 256-bit output; BLAKE2b with 64-bit words, 12 rounds, and 512-bit output.

[c]No data for SHA-3; numbers are for `keccakc512/1024` with 256- and 512-bit output sizes, respectively. These are the Keccak team's final submissions for SHA-3-256 and SHA-3-512.

V. Blockchained Post-Quantum Signatures Tailored to One-Time Keys

Most if not all few-time hash-based signature schemes make use of Merkle trees. The maximum number of messages a basic Merkle tree signature scheme can sign is 2^h, where h is the height of the tree. Also, all leaves (keys) should be computed during key generation in order to form the root. Due to the above, to construct a tree of height $h = 40$, key generation would be considered impractical, because we need to compute 2^{40} OTS keys and each OTS key internally requires many hash invocations (*i.e.*, 512 hash invocations with Lamport OTS or 67 for WOTS ($w = 16$) when using SHA256). The trick to keeping key generation time practical, while allowing for a large number of signatures is to use a multi-level tree.

BPQS is a simplified single-chain variant of the XMSS family protocols [5] which are literally an extension of the basic Merkle tree signature scheme (see Figure 1). BPQS can theoretically sign many times, but its design focuses on short and fast one-time signatures with the extra option to re-sign if and when needed. The above requirement is what a typical blockchain or DLT requires, as the use of one-time keys is recommended to preserve anonymity. However, a lot of things can go wrong, *e.g.*, a transaction might not go through or there might be a fork in the chain, in which case one should be able to sign more than one time without compromising security or freezing assets (see IOTA issues [21], [42]).

An additional, surprising benefit of BPQS is that it is also an ideal candidate for the opposite requirement; signing multiple times with the same key. This interesting property is due to the underlying graph-structure of blockchain and DLT systems that effectively allow many-time signatures at a minimal cost compared to other hash-based PQ solutions. This works by referencing the block (or transaction) in which the same BPQS key has been used in the past. In short, only a small part of the new signature is required to be submitted and the rest of the path will be delegated to the previous transaction this key was used to sign. The latter enables us to complete the full path to the advertised root BPQS key. Actually, because previous transactions are verified on the ledger already, verifiers do not even need to validate the rest of the path, as it was inherently verified in the past. This characteristic makes BPQS very useful for notary-based DLTs, such as Corda [22] and Fabric [43], as the notary nodes normally sign transactions with the same *known* key.

A. BPQS Scheme

BPQS requires an underlying OTS scheme. Although *any* OTS solution could in theory be applied, our scheme shares logic with the XMSS protocol family, hence the selection of the WOTS [19] variant, use of L-Trees and generation of bitmasks (blinding masks) define the security assumptions and proofs, similarly to XMSS [5], its multi-level version XMSS^{MT} [44] and XMSS-T [27]. Also, according to [39], collision resistance is actually cheaper using quantum algorithms, and thus similarly to the Gravity SPHINCS [33] scheme, bitmasks and L-Trees might be omitted.

One could state that BPQS is a subset of XMSS tailored to fast first-time signatures. The main difference is that XMSS overcomes the limitation to one message per key by using hash trees which reduce the authenticity of many OTS verification keys to one public XMSS root key. In contrast, BPQS utilises a chain of small 2-leaf Merkle trees. Geometrically, XMSS grows in both width and height (see Figure 1), while BPQS grows on chain height only (see Figure 2). All in all, we stress that BPQS is a generic blockchained construction, where blocks are "tiny" Merkle trees, meaning that it can be parameterised according to the requirements of the application. In case blinding masks are applied, their deterministic generation should follow the same logic with the corresponding XMSS family scheme.

There are 2 basic building blocks of BPQS:

- BPQS-FEW, which strictly supports few-time signatures and is depicted in Figure 2 (left),
- BPQS-EXT, which theoretically can be extended to support many-time signatures, see Figure 2 (right).

In BPQS-FEW, all keys are precomputed during key generation, the penalty for each extra signature is just 1 extra hash output, but it cannot be extended to practically support "unlimited" signatures.

On the other hand, BPQS-EXT initially requires only two OTS keys and in contrast to BPQS-FEW, the left leaf in each 2-leaf Merkle tree is an OTS fallback key that can be used to sign the next signature block when required. Unfortunately, the extensibility property comes with the cost of requiring one extra WOTS key per new signature.

The full BPQS scheme combines both BPQS-FEW and BPQS-EXT in a way where the last leaf in the chain of BPQS-FEW is a BPQS-EXT fallback key. This trick allows us to

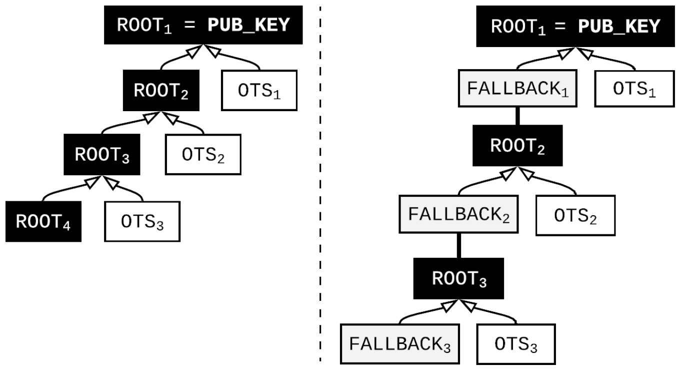

Fig. 2: BPQS-FEW (left), a few-time signature scheme. BPQS-EXT (right), a linearly extensible many-time signature scheme.

convert the few-time variant to a many-time one. Actually, BPQS-EXT can be considered a special case of BPQS, in which there is no initial BPQS-FEW chain.

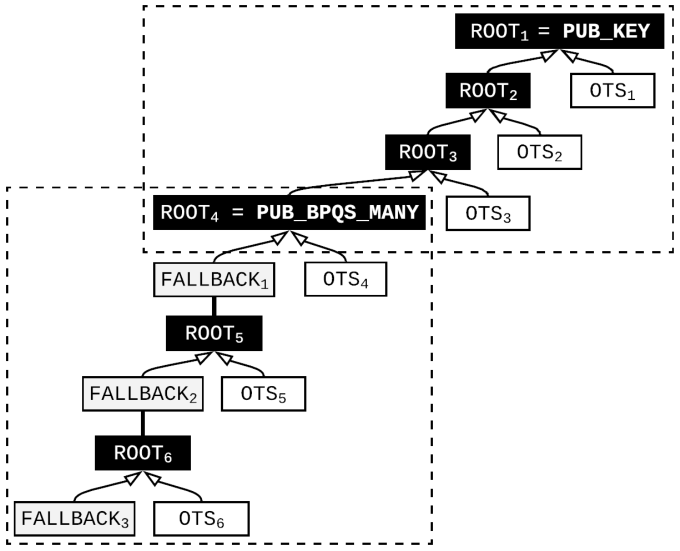

Fig. 3: Full BPQS protocol, with a height-3 BPQS-FEW top-level and a BPQS-EXT fallback key to allow for extensibility.

Parameters for BPQS include:

- the WOTS variant used (*e.g.*, WOTS-T [27]),
- the Winternitz parameter (*e.g.*, $w = 16$), which defines the base at which the initial hash is interpreted. Similarly to XMSS [5], w defines the actual size of each WOTS chain, which in turn affects signature size. Note that there is no consistency on the interpretation of the Winternitz parameter in the literature. For instance, in LMS [45] it is defined as 2^w and thus $w_{\text{BPQS}} = 16 = 2^4$ would be equivalent to $w_{\text{LMS}} = 4$,
- the underlying hash function (*i.e.*, SHA384),
- the number of precomputed OTS keys, meaning the initial height (*e.g.*, $h = 4$).

B. BPQS Mixed

The extensibility property of BPQS enables various custom constructions. BPQS can be used as a building block to convert any hash-based signature scheme into a {first or few}-time optimised one. For instance, in Figure 4, BPQS-FEW is used for the first (shorter) signatures and then it fallbacks to another PQ scheme. Although in the depicted approach the key-pair of the fallback (other) PQ scheme should be a-priori known and precomputed, one could use the BPQS-EXT in a similar fashion, so that this is not necessarily a requirement and the "other" PQ key will be generated only after the few-time signatures are exhausted. Moreover, if the "other" PQ scheme is stateless, such as SPHINCS, the final protocol is literally a "start stateful then go stateless" scheme.

It should be emphasised that the "other" PQ scheme might be another BPQS scheme, so one could eventually create a chain of different BPQS schemes. The latter would result in shorter signatures versus just extending it with BPQS-EXT each time.

With regards to the "Other PQ Key Params" shown in Figure 4, it is important that some schemes are required to publish bitmasks (or a seed in XMSS-T [27]) as part of the initial advertised public key. Otherwise, it would allow an adversary to select the seed/bitmask in a forgery. However, if BPQS uses a hash function with a bigger output (*e.g.*, SHA384 or SHA512) this might not be necessary, because the provided security-level against potential quantum collision attacks would still be enough to prevent such attacks.

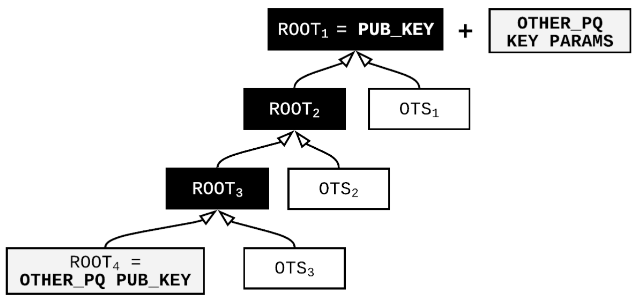

Fig. 4: A versatile BPQS protocol (BPQS-VERS1), with a BPQS-FEW top-level of height 3, in which the last root is the public key of another PQ scheme, such as XMSS^{MT} [44] or SPHINCS^{+} [32].

C. Combined PQ Schemes

As already mentioned, BPQS can fallback to another PQ scheme whenever required. By applying a similar logic, Figure 5 shows various custom models for combining multiple PQ schemes into one. The approach is very simple, but allows for very useful constructions, such as a "Stateful and Stateless" scheme in Figures 5 A and B, or a "Stateful with Stateless Fallback" scheme in Figure 5 C. The latter provides a solution to clustered environments in which multiple nodes require consensus over signature states, but a fallback mechanism is a prerequisite for the system to stay functional if consensus fails for any reason.

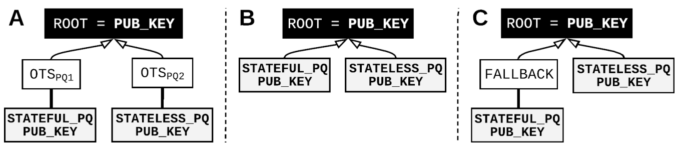

Fig. 5: Various recommended designs using a parallel BPQS logic to combine multiple schemes into one concrete PQ solution. Note that if the underlying schemes require extra parameters, such as bitmasks, these should be published along with the root public key, similarly to Figure 4.

The three depicted approaches offer different flexibility when it comes to:

- choosing a balance between key generation time and size of signature,
- deciding whether to allow for picking of the underlying algorithms at a later time.

For instance, option B requires both PQ keys to be generated to form the to-be-advertised combined public key, whilst option A is practically a BPQS-EXT that will be used to sign the "upcoming" PQ schemes. Along the same lines, option C is a combination of A and B, but the left PQ scheme is not required to be a-priori selected and computed. Note that one could even combine two different stateful or stateless schemes together, *e.g.*, if needed for compatibility purposes, such as when using the same key in two different blockchains, one supporting the original SPHINCS-256 [23] and the other supporting a variation of it (or its standardised version when this becomes available).

VI. Experimental Results

This section presents a performance analysis of BPQS based on an extensive experimental evaluation and a comparison against a selection of state-of-the-art signature schemes. The schemes compared against were chosen due to their use and popularity in today's PKI and blockchain communities.

Our results are based on a prototype implementation of BPQS available at [46], which includes details on how to reproduce our benchmark. System specifications include an octa-core Intel Core i7-7700HQ @ 2.80GHz with 15.5GB RAM running Linux 4.13.0-38 and JRE 1.8.0_161. Regarding the scheme implementation, the standard JCE has been used for hash function invocations, while implementations for other schemes are based on the BouncyCastle (ECDSA, RSA, SPHINCS-256) [47] and the i2p (EdDSA) [48] libraries, respectively.

Performance Comparison

Table 2 compares the performance of various signature schemes, including BPQS for $w = 4$ and $w = 16$ using both SHA256 and SHA384 as underlying hash functions. The results reported are average running time in milliseconds over a pre-generated pool of random strings. The main reason for picking a list of messages rather than a single or a few messages is to avoid any bias that might occur on the signature and verification operations, due to the message-to-be-signed structure; actually, its hash digest.

Scheme	KeyGen	Sign	Verify
BPQS ($w = 4$, SHA256)	0.569	0.08	0.10
BPQS ($w = 4$, SHA384)	1.107	0.16	0.19
BPQS ($w = 16$, SHA256)	0.872	0.19	0.20
BPQS ($w = 16$, SHA384)	1.719	0.39	0.38
ECDSA SECP256K1 (SHA256)	0.10	0.34	0.25
Pure EdDSA Ed25519 (SHA512)	0.18	0.08	0.16
RSA3072 (SHA256)	561.1	5.39	0.17
SPHINCS-256 (SHA512)	0.69	144.5	1.76

Table 2: Time (*ms*) of key-pair generation, signing and verification for first message. Each scheme is annotated with the hash used for producing the input digest.

It is emphasised that our current BPQS implementation is not optimised for parallel processing and it does not cache intermediate results of WOTS hash iterations. It is well-known [33] that caching on key secrets can speed up signature processing by a large degree. In practice, because BPQS is best suited to applications that will sign only one or a few times, the path and number of OTS keys are relatively small and would easily fit in memory. If all full WOTS structures are stored in memory during key generation, signing is nothing more than some memory lookups and would not involve running any hash operations at all, excluding the required message-hash upfront – in many blockchain applications, transactions are already hash digests (Merkle tree roots) anyway. We also stress that the cache can be recomputed on demand from a small secret seed and it does not need to be stored in persistent memory, as it can be regenerated after a reboot.

Even without parallelisation and/or caching, BPQS compares favourably or very similarly to both classical and PQ digital signature algorithms. It is highlighted that the simplest form of BPQS (BPQS-EXT) has been used in the above comparisons, where two WOTS-family keys are generated, one to sign the first message and the other for the fallback operation. We expect that on average a blockchain key is used only once and additional signing is only required in rare and special circumstances, thus our comparisons focus on the first signing operation only. In practice, we expect that blockchain wallets will use both caching and parallelisation, boosting performance further.

With regards to the signature size, all BPQS modes outperform XMSS for the first $(log_2(h) - 1)$ signatures. BPQS signatures grow linearly with the number of times a key is reused, thus the length of the signature output is dynamic. It starts small and increases per additional signature. Parameters should be chosen to balance between initial key generation cost and signature size. In the best case, the size increases by the size of one hash output for each extra signature and in the worst case by the size of one WOTS [19].

In the majority of BPQS modes, the size of the first signature output is $|WOTS|$ + $|H|$, while for XMSS it is always $|WOTS| + h \times |H|$, where h is the Merkle tree height used. When one of the most common WOTS with $w = 16$ is used as the underlying OTS scheme, the size of each WOTS, denoted as $|WOTS|$, is equal to $67 \times |H|$, where $|H|$ is the digest size of the underlying hash function, *i.e.*, 256 bits for SHA256. Figure 6 depicts the signature size per extra signing for the following schemes:

- BPQS-FEW (Figure 2 left), with $h = 32$,
- XMSS [5], with $h = 5$ where h is the tree height,
- BPQS-VERS1 (Figure 4), with a fallback to XMSS, both with $h = 5$,
- BPQS-GEN-B (Figure 5 B), where the right side is BPQS-FEW and the left side is XMSS, both with $h = 5$.

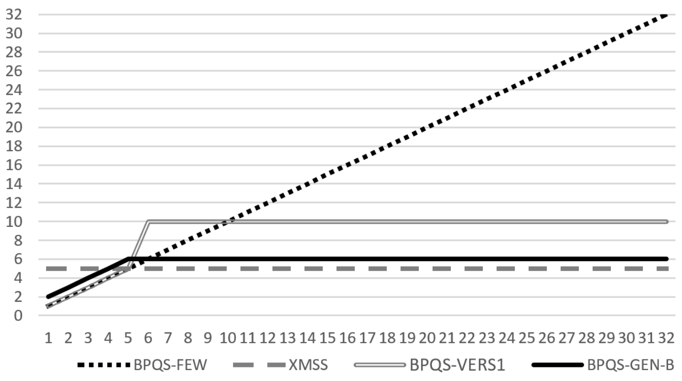

Fig. 6: Signature size comparison between different modes of BPQS and XMSS [5] supporting 32 OTS keys in total. Signature output for all schemes is $|WOTS|$ + x $|H|$, where x is the number of extra hashes required to complete the signature path. In this chart, *x-axis* is the number of signatures and *y-axis* is x.

VII. Conclusions and Future Work

In this work, we introduced BPQS and its extensions to support {one and few}-time optimised post-quantum signatures. We have also presented the security challenges that blockchains and DLTs will soon face and why pure OTS schemes are not recommended as a quantum-resistant replacement. As shown, BPQS compares favourably even against conventional non-quantum schemes such as RSA, ECDSA and EdDSA, while it provides more reliable quantum-security estimates because of its rooting in a secure cryptographic hash function.

Among others, the main features of the BPQS protocol are:

- shorter signatures, and faster key generation, signing and verification times than the XMSS [5] and SPHINCS [23] family PQ protocols when signing for one or few times, which is usually preferred in blockchain systems to preserve anonymity,
- it is computationally comparable to non-quantum schemes. One can take advantage of the easy-to-apply multiple hash-chain WOTS parallelisation and caching to provide almost instant signing and faster verification,
- its extensibility property allows for many-time signatures, while it can also easily be customised, so it can fallback to another many-time scheme if and when required,

- when used in blockchain and DLT applications, it can take advantage of the underlying chain/graph structure by referencing a previous transaction, in which the same key is reused. This could effectively mean that each new BPQS signature simply requires the effort of an OTS scheme, because the rest of the signature path to the root is in the ledger already and can be omitted,
- it could be used as a building block to implement novel PQ schemes such as a simultaneously "Stateful *and* Stateless" scheme, which might benefit clustered environments, where nodes can fallback to stateless schemes when consensus is lost. Additionally, such schemes can be used for forward and backward compatibility purposes or when requiring to reuse a key between two independent and incompatible blockchains.

The main drawback of the original BPQS protocol is that the size of its signature output increases linearly with the number of signatures. However, one can mitigate this by using a combined PQ approach or by utilising existing graph structures in blockchain applications. All in all, the customisation, caching and extensibility properties of BPQS make it an ideal candidate for blockchains and it could serve as a bridging protocol between stateless, stateful and other PQ schemes.

References

[1] P. W. Shor, "Algorithms for quantum computation: discrete logarithms and factoring", 35th FOCS, pp. 124-134, 1994.

[2] J. Proos, and C. Zalka, "Shor's discrete logarithm quantum algorithm for elliptic curves", Quantum Information & Computation, v.3 i.4, 2003.

[3] M. Mosca, "Cybersecurity in an era with quantum computers: will we be ready?", QCrypt, 2015.

[4] "The Quantum Countdown. Quantum Computing And The Future Of Smart Ledger Encryption", Long Finance, http://longfinance.net/DF/Quantum_Countdown.pdf, February 2018.

[5] J. Buchmann, E. Dahmen, and A. Hülsing, "XMSS – A Practical Forward Secure Signature Scheme Based on Minimal Security Assumptions", PQCrypto 2011: Post-Quantum Cryptography, pp. 117-129, 2011.

[6] J. Kelly, "A Preview of Bristlecone, Google's New Quantum Processor," Google Research Blog, https://research.googleblog.com/2018/03/a-preview-of-bristlecone-googles-new.html, March 2018.

[7] D-Wave, "Temporal Defense Systems Purchases the First D-Wave 2000Q Quantum Computer", D-Wave Press Release, https://www.dwavesys.com/press-releases/temporal-defense-systems-purchases-first-d-wave-2000q-quantum-computer, January 2017.

[8] T. F. Rønnow, Z. Wang, J. Job, S. Boixo, S. V. Isakov, D. Wecker, J. M. Martinis, D. A. Lidar, and M. Troyer, "Defining and Detecting Quantum Speedup", Science vol. 345, issue 6195, pp. 420-424, July 2014.

[9] L. K. Grover, "A fast quantum mechanical algorithm for database search", STOC, 1996.

[10] R. Anderson, and R. Brady, "Why quantum computing is hard-and quantum cryptography is not provably secure", arXiv:1301.7351, 2013.

[11] "CECPQ1 post-quantum cipher suite," Wikipedia article, https://en.wikipedia.org/wiki/CECPQ1, 2016.

[12] "The Post-Quantum PKI Test server", http://test-pqpki.com/, 2018.

[13] L. Chen, S. Jordan, Y-K. Liu, D. Moody, R. Peralta, R. Perlner, and D. Smith-Tone, "NISTIR 8105 Report on Post-Quantum Cryptography", NIST, 2016.

[14] NIST, "Post-Quantum Cryptography Standardization", https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Post-Quantum-Cryptography-Standardization, 2017.

[15] "Quantum Safe Cryptography and Security – An introduction, benefits, enablers and challenges", ETSI, http://www.etsi.org/images/files/ETSIWhitePapers/QuantumSafeWhitepaper.pdf, 2015.

[16] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev, "Scalable, transparent, and post-quantum secure computational integrity", IACR Cryptology ePrint Archive: Report 2018/046, 2018.

[17] T. Ruffing, P. Moreno-Sanchez, and A. Kate, "CoinShuffle: Practical Decentralized Coin Mixing for Bitcoin", ESORICS, 2014.

[18] P. Waterland, "The QRL Whitepaper", https://theqrl.org/whitepaper/QRL_whitepaper.pdf, 2011.

[19] A. Hülsing, "W-OTS+ – Shorter Signatures for Hash-Based Signature Schemes", IACR Cryptology ePrint Archive: Report 2017/965, 2017.

[20] S. Popov, "The Tangle", https://iota.org/IOTA_Whitepaper.pdf, 2017.

[21] "How bad is reusing an address?", IOTA forum, https://forum.iota.org/t/how-bad-is-reusing-an-address/1277, 2017.

[22] R. G. Brown, J. Carlyle, I. Grigg, and M. Hearn, "Corda: An Introduction", https://docs.corda.net/_static/corda-introductory-whitepaper.pdf, 2016.

[23] D. J. Bernstein, D. Hopwood, A. Hülsing, T. Lange, R. Niederhagen, L. Papachristodoulou, M. Schneider, P. Schwabe, and Z. Wilcox-O'Hearn, "SPHINCS: practical stateless hash-based signatures", EUROCRYPT 2015, pp. 368-397, 2015.

[24] L. Lamport, "Constructing digital signatures from a one-way function", Technical Report CSL98, SRI International, 1979.

[25] R. C. Merkle, "A Digital Signature Based on a Conventional Encryption Function", CRYPTO 1987 pp. 369-378, 1987.

[26] D. Bleichenbacher, and U. Maurer, "On the efficiency of One-Time Digital Signatures", ASIACRYPT, 1996.

[27] A. Hülsing, J. Rijneveld, and F. Song, "Mitigating Multi-Target Attacks in Hash-based Signatures", PKC 2016 pp. 387-416, 2016.

[28] A. Perrig, "The BiBa one-time signature and broadcast authentication protocol", 8th ACM Conference on Computer and Communication Security, pp. 28-37, 2001.

[29] L. Reyzin, and N. Reyzin, "Better than BiBa: Short One-time Signatures with Fast Signing and Verifying", ACISP 2002, pp. 144-153, 2002 .

[30] J.Buchmann, E. Dahmen, E. Klintsevich, K. Okeya, and C. Vuillaume. "Merkle Signatures with Virtually Unlimited Signature Capacity", ACNS, 2007.

[31] P. Kampanakis, and S. Fluhrer, "LMS vs XMSS: Comparion of two Hash-Based Signature Standards", IACR Cryptology ePrint Archive: Report 2017/349, 2017.

[32] D. J. Bernstein, C. Dobraunig, M. Eichlseder, S. Fluhrer, S-L. Gazdag, A. Hülsing, P. Kampanakis, S. Kölbl,

[33] J-P. Aumasson, and G. Endignoux, "Improving Stateless Hash-Based Signatures", IACR Cryptology ePrint Archive: Report 2017/933, 2017.

[34] S. Gueron, and N. Mouha "SPHINCS-Simpira: Fast Stateless Hash-based Signatures with Post-quantum Security", IACR Cryptology ePrint Archive: Report 2017/645, 2017.

[35] S. Kölbl, M. Lauridsen, F. Mendel, and C. Rechberger, "Haraka v2 – Efficient Short-Input Hashing for Post-Quantum Applications", IACR Cryptology ePrint Archive: Report 2016/098, 2016.

[36] Federal Information Processing Standards Publication 180-4, "Secure Hash Standard (SHS)", Information Technology Laboratory, National Institute of Standards and Technology, March 2012.

[37] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche, "The KECCAK SHA-3 submission, Version 3", 2011.

[38] M. Amy, O. Di Matteo, V. Gheorghiu, M. Mosca, A. Parent, J. Schanck, "Estimating the Cost of Generic Quantum Pre-image Attacks on SHA-2 and SHA-3", IACR Cryptology ePrint Archive: Report 2016/992, 2016.

[39] D. J. Bernstein, "Cost analysis of hash collisions: Will quantum computers make SHARCS obsolete?", SHARCS 2009, 2009.

[40] "Measurements of hash functions, indexed by machine", eBACS: ENCRYPT Benchmarking of Cryptographic Systems, http://bench.cr.yp.to/results-hash.html, accessed: 21 February 2018.

[41] M-J. Saarinen, and J-P. Aumasson, "The BLAKE2 Cryptographic Hash and Message Authentication Code (MAC): IETF RFC 7693.", Internet Engineering Task Force. DOI: 10.17487/RFC7693, 2015.

[42] "IOTA ERROR: PRIVATE KEY REUSE DETECTED", IOTA github, https://github.com/iotaledger/wallet/issues/928, 2018.

[43] C. Cachin, "Architecture of the Hyperledger Blockchain Fabric", https://www.zurich.ibm.com/dccl/papers/cachin_dccl.pdf, 2016.

[44] A. Hülsing, S-L. Gazdag, D. Butin, and J. Buchmann, "Hash-based Signatures: An Outline for a New Standard", NIST Workshop on Cybersecurity in a Post-Quantum World, 2015.

[45] D. McGrew, and M. Curcio. "Hash-Based Signatures", https://datatracker.ietf.org/doc/draft-mcgrew-hash-sigs, accessed: April 2018.

[46] "BPQS library", https://github.com/corda/bpqs, accessed: April 2018.

[47] "Bouncy Castle Crypto APIs", v2.1.1, release: 1.59, 2017.

[48] "EdDSA-Java", v0.2.0, https://github.com/str4d/ed25519-java, 2018.