Shafarenko *Cybersecurity (2024)7:34*
https://doi.org/10.1186/s42400-024-00225-9

Cybersecurity

RESEARCH

Open Access

Winternitz stack protocols for embedded systems and IoT

Alex Shafarenko[1*]

Abstract

This paper proposes and evaluates a new bipartite post-quantum digital signature protocol based on Winternitz chains and an oracle. Mutually mistrustful Alice and Bob are able to agree and sign a series of documents in a way that makes it impossible (within the assumed security model) to repudiate their signatures. The number of signatures supported by a single public key is still limited, though by a large number. However, the security of the signature scheme is not diminished by repeated application, so when the capacity of a public key is exhausted the last transaction can be used to agree a new key. Some ramifications are discussed, security parameters evaluated and an application area delineated for the proposed concept.

Keywords Signature protocol, Hash-based signature, Post-quantum

Introduction

Our focus is on supporting multivendor embedded devices that require guaranteed nonrepudiation. Such devices often occur in automotive, aerospace and other safety-critical applications, as well as in all kinds of medical technology. Authentication and integrity-control techniques that utilise symmetric ciphers are based on sharing a confidential key. The same key is used for signing messages and validating them, so the sender can always repudiate the message by claiming that it was formed by the other party that shares the key. Public-key cryptography does not quite solve this problem. First of all, it is vulnerable to quantum attacks and cannot be relied on in a future-proof technology. Secondly, and perhaps more importantly from a practical point of view, signature calculation and verification require a volume of computations that can be too large for an embedded device operating on a tight power budget.

We attempt to address both issues by proposing a hash-based signature. Such signatures exploit the one-way nature of a cryptographic hash to create an effective public-key/private-key pair. Large random integers are used as the private key, and their hashes are published to form the public key. When a new message requires a signature, the signer reveals certain integers of the private key. Their choice uniquely identifies the message, and since the only principal who can show the pre-image of a hash is the one who created that hash in the first place, a collection of pre-images reliably linked to a message can serve as a signature.

There exist quite a few hash-based signature schemes that are well developed and understood, see "Related work" section for related work. We draw our inspiration from Winternitz's idea to apply a hash to a random seed repeatedly to create a chain, and Reyzin and Reyzin's idea of random index sets (multisets in our approach) generated by a hash and used to index an array of key-pairs. However, our proposal differs from the related work by the fact that we build a signature stack using an array of long Winternitz chains (we call this a Winternitz fabric), and we push fixed cardinality multi-set frames on it rather than variable-cardinality subsets, see "Winternitz stack protocol" section. This way we achieve security that does not diminish under repeated application, so ours is not a *few-time* signature scheme, but in fact a constant

*Correspondence:
Alex Shafarenko
a.shafarenko@herts.ac.uk
[1] University of Hertfordshire, Hatfield AL10 9AB, UK

security scheme, even though the capacity to sign is limited by the length of the fabric. Having said that, for a very modest storage capacity (single gigabytes), a million signatures can be accommodated without recalculation. Storage capacity can be traded in for hash recalculations and under realistic assumptions the storage requirements can be reduced to megabytes without significantly altering the amount of work required for signing, see "Practicalities" section. We show that 256-bit security (128-bit post-quantum) is easily achieved by our scheme. The computational burden on the parties after the protocol launch amounts to a few tens of hash calculations by the verifier (same as the original HORS by Reyzin and Reyzin 2002) and only one or two by the signer—a negligible amount compared to the computations involved in a public-key signature scheme such as ECDSA.

The most unusual feature of our solution is its ability to support a mutual signature of two parties using a single public key by repurposing the confirmation pre-image for signing the verifier's approval, see "MAWS protocol" section. We propose an extreme version of this mutual protocol, which we call reverse Winternitz stack (RWS), where Alice signs nothing but messages received from Bob, see "Reverse Winternitz stack (RWS) protocol" section. As a result Alice could become Bob's public notary without either of them requiring trust. Indeed if a third party trusts Alice (authenticated by her public key) not to collude with Bob and not to deny him service by breaking the protocol, Bob can safely sign his messages by running RWS with her for the third party's use. If Alice tries to impersonate Bob to sign a message, Bob will be able to prove the signature false, but if the message is in fact genuinely signed by Bob, then Alice is also safe, since Bob will not be able to repudiate his signature. Alice and Bob have instant assurances by the protocol, but any proof for a third party requires dumping Bob's stack. The stack contains digests of all documents that Bob's ever signed with Alice since the establishment of the current public key, as well as additional protocol data the size of that key (hundreds of KB). This gives rise to a small communication requirement in the order of 1Mbyte. In most scenarios involving a guarantee of nonrepudiation, a third-party proof is only required after a major event (car breakdown, aircraft malfunction, etc.) to adjudicate on the cause of the event in a multivendor environment.

Another remarkable feature of RWS is its communication asymmetry. Bob receives of the order of 1 Kb of data from Alice, but his transmission requirements are limited to only 64 bytes per signature, same as it would be for the 256-bit ECDSA. In an IoT situation, where messages are communicated over long distances via a low bit-rate radio, *transmission* requires much power to radiate a strong enough signal to reach a network hub, while *reception* involves only digital signal processing. Also the maximum radiated power and the transmitter duty cycle of an IoT radio is limited by law to enable public use without harmful interference. By contrast, IoT network hubs are allowed a more powerful transmitter (up to a factor of 10), higher duty cycle (again a factor of 10) and an elevated full-size antenna to be able to transmit much greater volumes of data. The RWS protocol nicely matches this asymmetry.

The main contributions of the paper are as follows:

- The idea of Winternitz stack and an analysis of its security properties
- Three signature protocols based on a Winternitz stack with nondecreasing security of a large but limited number of signatures
- Analysis of the protocol resource footprints

The next two sections present the basic principles, notations and some security properties of the Winternitz stack. "Winternitz stack protocol", "MAWS protocol" and "Reverse Winternitz stack (RWS) protocol" sections describe the signature protocols. "Application of RWS to Internet of Things" section discusses possible applications, "Related work" section presents related work and finally there are some conclusions.

Principles of Winternitz stack signature

We begin with the standard definition of Winternitz chain:

Definition 1 For a cryptographic hash-function $H(x)$ and some arbitrary r_0 of the same bit-length as the image of H, the sequence

$$r_{i+1} = H(r_i), \text{ for } 0 \le i < N$$

is called a length-N *Winternitz chain*.

Let us bring several chains together.

Definition 2 (*Winternitz fabric*) An indexed family of length-N Winternitz chains $r_i^{[k]}$, where k is the index, $0 \le k < w$, w is a power of 2, and $0 \le i < N$, is called a *Winternitz (w, N)-fabric*, or just fabric for short. The constant w is called the *width* and N, the length of the fabric. The indexed family $(E^{[k]})$, where $E_k = r_N^{[k]}$ where $0 \le k < w$ is called the *edge* of the fabric. When referring to the edge as a whole we will omit the index: E, or $E(F)$ for the edge of a fabric F.

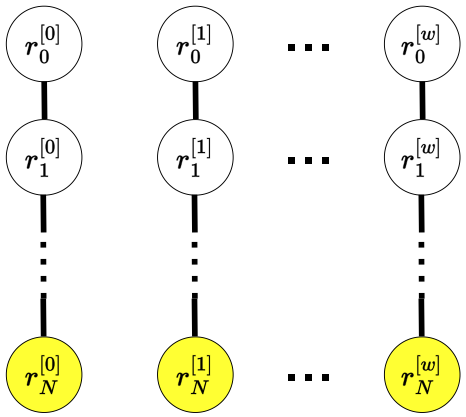

Fig. 1 Winternitz fabric. The shaded nodes represent the fabric edge. A vertical line followed down connects a value with its hash image

See Fig. 1 for an illustration. The width of a fabric is constrained to a power of 2, but the length of the chain is arbitrary. For practical purposes chain lengths of the order of one million or less should satisfy most demands for a digital signature. Notice that due to the hardness of the second preimage problem, at most one (w, N)-fabric can be produced given an arbitrary width-w edge E for any practical N.

Consider a length-d sequence of binary strings D_j, $0 \le j < d$, which we will call *documents*. Let a function

$$\Omega^{[\kappa,w]} : \mathcal{B} \to \mathcal{M}_{\kappa,w}$$

be a random oracle, where $\mathcal{B}$ is a set of arbitrary-length binary strings and $\mathcal{M}_{\kappa,w}$ is a set of all cardinality-κ multisets of integers taken from the range $[\![0, w-1]\!]$. Assume the oracle is such that the presence of those integers in the multiset is uncorrelated. A multiset $M \in \mathcal{M}_{\kappa,w}$ of this kind can be defined using its characteristic function $\chi_M : [\![0, w-1]\!] \to [\![0, \kappa-1]\!]$, such that for any $x \in [\![0, w-1]\!]$ x occurs in M $\chi_M(x)$ times. For convenience, define $\omega : \mathcal{B} \times [\![0, w-1]\!] \to [\![0, \kappa-1]\!]$ to be a function such that $M = \Omega^{[\kappa,w]}(b)$ iff $\omega(b,k) = \chi_M(k)$ for any $b \in \mathcal{B}$ and $0 \le k < w$. Clearly, for any $b \in \mathcal{B}$,

$$\sum_{k=0}^{w-1} \omega(b,k) = \kappa. \tag{1}$$

Cardinality κ is a security parameter. We assume it is fixed and will discuss the choice of it later on. In the sequel we will not show the dependency of any variables of interest on κ explicitly and will not use explicit upper indices of Ω if the omission does not lead to ambiguity.

Definition 3 For an integer $d > 0$, a depth-d signature stack over a Wintenitz (w, N)-fabric $(r_i^{[k]})$ is a pair of indexed families $(D,\ T)$, where $D = (D_j)_{j \in [\![0,d-1]\!]}$ is a sequence of documents, and $T = (r^{[k]}_{N-\sigma(k)})_{k \in [\![0,w-1]\!]}$ is the top of the stack, with the function $\sigma : [\![0, w-1]\!] \to [\![0, N-1]\!]$ defined thus:

$$\sigma(k) = \sum_{m=0}^{d-1} \omega\Big(\|_{j=0}^{m} D_j, k\Big), \tag{2}$$

provided that $\sigma(k) \le N$ for all k in its domain. (The double vertical bar denotes bit-string concatenation.)

A depth-0 signature stack is the pair $(\emptyset, (r_N^{[k]})_{k \in [\![0,w-1]\!]})$.

Corollary 1 *For any depth-d signature stack* $(D,\ T)$ *over a Wintenitz* (w, N)*-fabric,*

$$\sum_{k=0}^{w-1} \sigma(k) = d\kappa.$$

Proof Sum in k both sides of the equation for σ in Definition 3, make the summation in k innermost in the right-hand side and use Eq. 1. □

Fabric capacity

Typically what is signed is not the actual content but its cryptographic digest, so D_j are usually hash-images of the actual documents to be signed. Also, to prevent a replay attack, the original content typically contains a random nonce. Under such assumptions the family (D_j) is a collection of random values which makes the oracle output not only random but also, with a probability very close to 1, free from repetitions. Let us evaluate the capacity of the fabric to carry a signature stack of a large depth d.

First visualise the fabric as w vertical rods that balls can be slid unto, see Fig. 2. Each document D_j causes balls to be slid on some of the rods according to ω, the total number of balls being κ. Their distribution over the rods is random and uncorrelated. In particular, it is possible but not very probable that some balls for a given document will be slid on the same rod. The distribution $\sigma(k)$ is the result of repeating "the sliding of balls" d times using a total of $d\kappa$ balls. Choose one rod at random and observe that the probability for a ball to end up taking that rod is $1/w$. The number of balls on the rod after $d\kappa$ balls have been randomly distributed between the rods is governed by the binomial distribution, which gives us the obvious expectation $E = d\kappa/w$ and the standard deviation

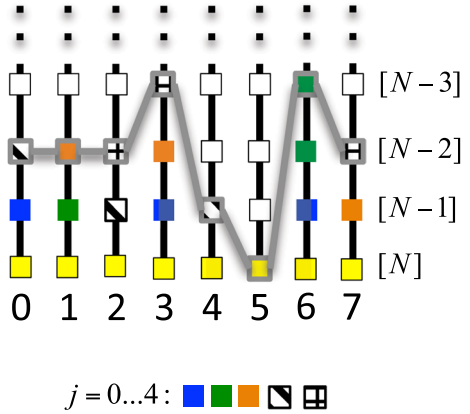

Fig. 2 An example of a depth-5 signature stack over a Winternitz (8, *N*)-fabric using a random oracle with $\kappa = 3$. Empty boxes represent unused members of the fabric. The top of the stack *T* is highlighted in grey and the fabric edge in yellow. The document family is not shown

$$\Delta = \sqrt{\frac{d\kappa}{w}\left(1 - \frac{1}{w}\right)} \approx \sqrt{\frac{d\kappa}{w}},$$

for large d. By Central Limit Theorem of statistics the distribution becomes close to Gaussian at large d. The rule of thumb is that fluctuations of a Gaussian random value very rarely exceed 6Δ and so we calculate, for large N that we are interested in,

$$\frac{d_{\max}\kappa}{w} \lesssim N - 6\sqrt{N}.$$

The exact point at which the fabric will prove too small to support the signature stack over it depends on the documents (D_j), but since each document only slides κ balls on the rods, and since we will always use wide fabrics ($\kappa \ll w$), the process can be stopped very close to that point. For estimates we should neglect $\sqrt{N}$ compared to N and use

$$d_{\max} \sim \frac{wN}{\kappa}. \tag{3}$$

Oracle entropy

Next let us explore security properties of a signature stack. The lynchpin of security here is the fact that a random oracle $\Omega(D)$ makes the problem of a second preimage unfeasibly hard, provided that its codomain is a large set. An attacker trying to find some D' that has the same image as D, $\Omega(D') = \Omega(D)$ will have to make a number of attempts commensurate with $|\mathcal{M}_{\kappa,w}|$, which could be astronomically large. We find from elementary combinatorics that for a fabric of width w, a random oracle with the security parameter κ will yield one out of a possible

$$G = \binom{w + \kappa - 1}{\kappa}$$

multisets of cardinality κ with equal probability. For practical reasons, which we will explain later, we are interested in a small κ of the order ten, while we are willing to consider fabrics of a width of a few thousand. Expanding the above for $\kappa \ll w$ and keeping the first nonvanishing term in $1/w$ we obtain

$$G = \frac{(w + \kappa - 1) \times \cdots \times w}{\kappa!} \sim \frac{w^\kappa}{\kappa!}\left(1 + \frac{\kappa(\kappa - 1)}{2w}\right). \tag{4}$$

Note that this is an estimate from below as all higher-order terms are positive. We are interested in the entropy of the oracle, which equals the binary logarithm of G. To get some idea how large it can be for $\kappa^2 \lesssim w$ we neglect the term in brackets and use Stirling's formula for the factorial, which is accurate to within 1% even for κ as small as 10:

$$\log_2 G \approx \kappa \log_2(we/\kappa) - \frac{1}{2}\log_2(2\pi\kappa).$$

Consider a fabric of width $w = 4096$ and an oracle with $\kappa = 31$ and find that $\log_2 G$ exceeds 259. Notice that it is marginally better than the security of SHA-256 with respect to a second preimage attack.

HORS oracle

A true random oracle is not feasible, but it can be approximated very well using the trick invented by Reyzin and Reyzin as they proposed their HORS scheme (Reyzin and Reyzin 2002). Specifically we take a standard hash of the argument D and pare it down to $\kappa \log_2 w$ bits.[1] In our numerical example this would be $31 \times 12 = 372$ bits. A hash of this length is easy to compute by application of SHA-512, taking the first 372 bits of the result. All that remains is to partition the bit string into 12-bit chunks, interpret them as unsigned binary integers, and to collect them into a multiset, which will represent the value of $\Omega(D)$. We call this implementation a *HORS oracle* and formally define it next.

Definition 4 For any $w = 2^q$ with some positive integers q and κ, a HORS oracle is a function $\eta^{[\kappa,w]} : \mathcal{B} \to \mathcal{M}_{\kappa,w}$ defined as follows:

[1] Remember that the width of a fabric is required to be a power of 2 by the above definition.

$$\eta^{[\kappa,w]}(x) = \Psi_{\kappa,w}(\hat{H}(x) \mod w^\kappa),$$

where $\Psi_{\kappa,w} : \mathcal{B}^{[\kappa q]} \to \mathcal{M}_{\kappa,w}$ is the multiset of κ digits of the κq-bit integer argument written in the base-w positional number system and $\hat{H} : \mathcal{B} \to \mathcal{B}^{[u]}$ is a cryptographic hash function producing a u-bit hash value, $u \geq \kappa q$.

Strictly speaking, a HORS oracle would not be a random oracle even if we disregarded the difference between a cryptographic hash $H(x)$ and a genuine random function. The random oracle makes a random selection of a multiset from the codomain $\mathcal{M}_{\kappa,w}$; any $X \in \mathcal{M}_{\kappa,w}$ is selected with the same probability. The HORS oracle (again, ignoring the non-random nature of the hash) does not select a multiset; it selects κ random numbers from the interval $[\![0, w-1]\!]$, which may or may not be pairwise distinct. If they are, then the multiset is a set of cardinality κ and its statistical weight in the codomain of $\hat{H}$ is $\kappa!$. If the collection of numbers has a single pairwise collision, its statistical weight is only half as much. Multiple collisions degrade the statistical weight even further. Consequently, even if the image of $\hat{H}(x)$ is evenly spread over the codomain, the entropy of the HORS oracle may be quite different from the estimate given by Eq. 4. How much different?

It should be noted that the number of proper multisets (i.e. multisets that are not sets) is small compared to the number of sets in the oracle's codomain:

$$G_+ = \binom{w+\kappa-1}{\kappa} - \binom{w}{\kappa} \sim \frac{w^\kappa}{\kappa!}\frac{\kappa(\kappa-1)}{w} \sim R\gamma,$$

where we kept the first nonvanishing term in the expansion in the birthday[2] parameter $\gamma = \kappa^2/w$. The factor $R = w^\kappa/\kappa!$ corresponds to the number of distinct sets (ignoring the collisions) swept by the indices as they independently cover their value interval $[\![0, w-1]\!]$. The most frequent proper multiset has one binary collision. There are

$$G_1 = w\binom{w-1}{\kappa-2} \sim \gamma\binom{w}{k} \sim \gamma R$$

of those, again keeping to the first nonvanishing term in γ. Comparing this with the expression for G_+ above, we find that the contribution of multiple collisions is higher order in γ. Indeed the next term corresponds to *two binary* collisions since there are a factor of κ/w fewer multisets with one tertiary collision. The statistical weight of the former is

$$\binom{w}{2}\binom{w-2}{\kappa-4} \sim \frac{1}{2}\gamma^2 R,$$

and can safely be neglected for small enough γ (in our example $\gamma \approx 1/4$) along with the rest of the higher-order terms. By contrast, the number of multisets that are sets can be approximated to the first order in γ as

$$G_0 = \binom{w}{\kappa} = R\Big(1 - \frac{\gamma}{2}\Big).$$

We can now construct an approximate probability distribution function (PDF) for multisets by assuming that a multiset is either a set or it has one binary collision:

$$f(s) = \begin{cases} 1/z & \text{if s is a set,} \\ 1/(2z) & \text{otherwise,} \end{cases}$$

where z is the normalising constant that satisfies the following:

$$G_0\frac{1}{z} + G_1\frac{1}{2z} = 1,$$

which gives us $z = R$. Finally, summing over all sets and proper multisets and keeping to the main order in γ we arrive at the entropy value

$$\begin{aligned} H &= -G_0\frac{1}{R}\log_2\frac{1}{R} - G_1\frac{1}{2R}\log_2\frac{1}{2R} \\ &= \log_2 R + \frac{\gamma}{2} \sim \log_2 G - \frac{\log_2 e - 1}{2}\gamma, \end{aligned}$$

where the last step is achieved by taking the binary logarithm of Eq. 4 and keeping to the first order in γ. Informally, proper multisets expand the alphabet of the HORS oracle output compared to just sets, thus increasing its entropy, however they make the PDF uneven and this reduces the entropy by almost the same amount. As a result, a HORS oracle with a moderately small γ has almost exactly the same entropy as the random oracle Ω provided that the hash function $\hat{H}$ is close to ideal.

Proof of stack security by reduction

The previous section assumed the framework of the random oracle model (ROM) (Bellare and Rogaway 1993) as it identified the hash function $\hat{H}(x)$ with a random oracle for the purposes of computing the entropy of $\eta^{[\kappa,w]}$. That is the best we can do given that the statistics of a practical cryptographic hash are unknown but generally believed to be close to ROM. However, in analysing the security of the Windernitz stack, on which all subsequent protocols rely, we prefer to remain within the Standard Model since the irreducible assumptions about the relevant hash functions (and we intend to use two generally different hashes) are weaker than the ROM would impose.

[2] So named as it defines the probability of collision in the birthday paradox.

This section presents the key security property (Lemma 1) and its reduction to the those assumptions, which follow below.

Assumption 1 The hash function $\hat{H}$ used to construct HORS oracle is one-way (preimage and second preimage resistant Rogaway and Shrimpton 2004).

Assumption 2 The hash function H used to construct the Winternitz fabric is preimage-resistant.

The key property that provides security to the Winternitz stack is that the HORS oracle is preimage and second-preimage resistant, which makes it hard to fit a random digest to the top of the stack to forge the signatures of the rest, or even to substitute a new digest for an old without changing the rest of the stack content.

By contrast, in constructing a Winternitz fabric over which the stack is placed, only simple preimage resistance is required of the chain hash-function. Indeed, an attacker may find a second preimage and claim that it is the next node of the chain in an attempt to split it. However, to succeed the attacker would then have to find the preimage of that second image to continue the chain; this would fail if H is preimage resistant. The fabric is never used up since the shape of the stack top is unpredictable and it may run out of fabric for the next document, so some fabric is left unused using a conservative estimate, and then the principals switch to a new fabric. Consequently the adjudicator is in a position to ask the fabric owner to prove the top of the stack is genuine by exposing the next node of the chain about which the suspicion of chain-split has arisen.

Let us now redefine $\omega(b,k)$ (which we defined earlier based on the ideal oracle $\Omega^{[\kappa,w]}$) in terms of our practical HORS oracle $\eta^{[\kappa,w]}$; this will not lead to a confusion since the sequel has no reference to Ω. Specifically, $\omega(b,k)$ from now on will be a function such that $M = \eta^{[\kappa,w]}(b)$ iff for all $0 \le k < w$, k occurs in M $\omega(b,k)$ times, for any multiset M and string $b \in \mathcal{B}$.

Proposition 1 *For any depth-d signature stack* $(D,\ T)$ *over a fabric, it is computationally hard to find a family* $D' \neq D$ *such that* (D',T) *is a depth-d signature stack over the same fabric.*

Proof Essentially one would have to solve the following set of simultaneous equations for (D'_j):

$$\sum_{m=0}^{d-1} \omega\left(\|_{j=0}^{m} D'_j, k\right) = \sum_{m=0}^{d-1} \omega\left(\|_{j=0}^{m} D_j, k\right) = \sigma_k\ , 0 \le k < w.$$

The left-hand side is the sum of d terms. If any document D_l is changed it would be computationally difficult to avoid change in all terms of the sum for which $l \le m < d-1$, since according to Assumption 1, the hash function in η, namely $\hat{H}(x)$, is second-preimage resistant. Clearly the least work is required when $l = d-1$, i.e. only the last document is changed. Still, one second preimage would need to be found. Alternatively, one could build a stack with a depth $d-1$ using any digests and then find the digest corresponding to the last multiset to be added to result in the same stack edge as defined by σ_k. That is also computationally hard due to Assumption 1 (preimage resistance). □

The values σ for a stack can be derived from its T by computing $\beta(T_k, E_k)$ where $\beta(x,y)$ for $x \neq y$ is the least positive integer i such that

$$\underbrace{H(H(\ldots H}_{i \text{ times}}(x)\ldots)) = y.$$

If $x = y$, we define $\beta(x,y) = 0$. It should be noted that β is a partial function $\mathcal{B} \times \mathcal{B} \nrightarrow \mathbb{N}$, since the value of i that satisfies its definition may not exist. If i does exist, it is less than the fabric length, which makes the definition constructive.

Since $T_k = r^{[k]}_{N-\sigma(k)}$, $\beta(T_k, E_k) = \sigma(k)$. From Corollary 1 for a valid stack $(D,\ T)$ we have:

$$\sum_{k=0}^{w-1} \beta(T_k, E_k) = d\kappa. \tag{5}$$

We will use (D,σ) and $(D,\ T)$ interchangeably where it does not create a confusion.

Definition 5 (Substack) For a depth-d stack $S = (D,\sigma)$ over a $(w,\ N)$-fabric, a depth-d' stack $S' = (D',\sigma')$ over the same fabric is a substack of S if for all $0 \le k < w$, $\sigma'(k) \le \sigma(k)$.

Proposition 2 *Consider a depth-d′ substack* $S' = (D',\sigma')$ *of a depth-d stack* $S = (D,\sigma)$. *If* $d' = d$, *then* $\sigma' = \sigma$.

Proof Proof by contradiction. Assume $\sigma' \neq \sigma$, and since for all $0 \le k < w, \sigma'(k) \le \sigma(k)$, then $(\exists k_0)\sigma'(k_0) < \sigma(k_0)$. But then

$$\sum_{k=0}^{d'-1} \sigma'(k) < \sum_{k=0}^{d-1} \sigma(k).$$

By Corollary 1 we have

$$\kappa d' < \kappa d,$$

which is a contradiction since $d' = d$. □

Notice that Proposition 2 does not generalise down. If $d' < d$, especially when $d' \ll d$, almost any documents (D_j) will place the top of the substack lower on the fabric than the larger stack's top. Indeed to make the substack tall with a small d' would require a document whose distribution ω over the fabric is restricted to very few values of k which is very improbable for a good approximation of a random oracle. Consequently there exists plenty of substacks of a given stack with a smaller depth.

Lemma 1 (Stack security) *If T is known to be the top of a depth-d stack (D, T) over a private fabric F with a public edge $E = E(F)$, the pair (E, T) is sufficient to find d and identify D. It is computationally hard for an adversary with no knowledge of the rest of F to produce an alternative $D' \neq D$, such that (D', T') with any T' is a valid depth-d stack over a fabric with the same edge $E(F)$.*

Proof To find d from T, use Eq. 5. Next, observe that knowledge of (D, T) is always sufficient to reconstruct all members of the fabric that the stack and any of its substacks occupy down to the edge E. Although given just that knowledge it is possible to construct a valid substack $(D' \neq D, T')$ for some arbitrary documents (D'_j), but according to Proposition 2 the substack would have to be of a depth less than d, which contradicts the premise of the Lemma. If $T' = T$, changing even a single D_j to $D'_j \neq D_j$ without changing T is computationally hard according to Proposition 1. Finally, if (D', T') is not a substack of (D, T), to produce T' one would require fabric elements that cannot be derived from T, and the premise states that the fabric is private. These unknown private elements cannot be obtained from the fabric elements at the top of the stack due to Assumption 2. □

Ancillary operations

Stack push

Signature stacks over a fabric are inherently sequential. It is possible to extend a stack to accommodate an extra document provided that the fabric is accessible.

Definition 6 Stack push p is a partial function $\mathbf{p} : \mathcal{D} \times \mathcal{S}_d \nrightarrow \mathcal{S}_{d+1}$, where $\mathcal{D}$ is, as before, a set of all finite binary strings and $\mathcal{S}_d$ is an indexed family of sets of all depth-d stacks over some fixed (w, N)-fabric $(r_k^{[i]})$:

$$\mathbf{p}(\delta, ((D_j)_{0\le j<d}, (r^{[k]}_{N-\sigma(k)})_{0\le k<w})) \triangleq ((D'_j)_{0\le j<d+1}, (T'_k)_{0\le k<w}),$$

where

$$D'_j = D_j \text{ for } 0 \le j < d,$$
$$D'_d = \delta,$$
$$\sigma'(k) = \sigma(k) + \omega\left(\|_{j=0}^{d} D'_j, k\right) \text{ for } 0 \le k < w,$$

and

$$T' = \left(r^{[k]}_{N-\sigma'(k)}\right)_{k\in[\![0,w-1]\!]} \text{for } 0 \le k < w,$$

provided that all such r exist in the fabric. Otherwise the result is undefined.

Observe that[3] at least for some k, $\sigma'(k) > \sigma(k)$. This means that the stack push always depends on unused members of the fabric and requires access to it.

Operations on indexed families

We require two ancillary operations on indexed families: addition and subtraction.

Definition 7 Let A and B be two indexed families $A = (A_i)_{i\in C}$ and $B = (B_i)_{i\in C}$ with indices from the same finite $C \subset \mathbb{Z}^+$. We define the *difference* $A - B$ as the indexed family $A - B = (A_i)_{i\in C^*}$, where

$$C^* = \{i \in C \mid A_i \neq B_i\}.$$

Definition 8 Let A and B be indexed families $A = (A_i)_{i\in C}$ and $B = (B_i)_{i\in C^*}$, where $C \subset \mathbb{Z}^+$ is some finite set and $C^* \subseteq C$. We define the *sum* $A + B$ as the indexed family $(Q_i)_{i\in C}$, where

$$Q_i = \begin{cases} B_i & \text{if } i \in C^* \\ A_i & \text{otherwise} \end{cases}.$$

Proposition 3 *For any two finite indexed families $A = (A_i)_{i\in C}$ and $(B_i)_{i\in C}$, where C is a finite index set,*

$$B + (A - B) = A.$$

Proof (by cases) For a given index value i, if $A_i \neq B_i$ then, by Definition 7, $(A - B)_i = A_i$ and i belongs to the index set of $A - B$. But if it does, then by Definition 8

$$(B + (A - B))_i = (A - B)_i = A_i.$$

Otherwise $B_i = A_i$ and, by Definition 7, the index value i does not belong to the index set of $A - B$. Then by Definition 8

[3] Ignoring the infinitesimal probability of a good approximation of a random oracle producing all-zeros (more than 300 zeros in our numerical example)

$$(B + (A - B))_i = B_i = A_i.$$

□

If two parties share a family of bit-strings B and at some point one needs to send to the other a similar family A which has many common members with B, it would be sufficient to communicate $A - B$, which has a much smaller index set, and then the receiving party will restore A by computing $B + (A - B)$.

Definition 9 Given an indexed family $X = (X_i)_{i\in I}$, where I is a finite index set $I \subset \mathbb{Z}^+$, and some element x, the *extension* of X with x is the family $X' = X \triangleright x$ such that $X' = (X'_i)_{i\in I'}$,

$$I' = I \cup \{\max_I i + 1\}$$

and

$$X'_i = \begin{cases} X_i & \text{if } i \in I \\ x & \text{otherwise} \end{cases}.$$

Validator

Definition 10 Consider a stack $S = (D, T)$, $D = (D_j)_{j\in[\![0,d-1]\!]}$ and $T = (T_k)_{k\in[\![0,w-1]\!]}$, over a width-$w$ fabric of sufficient length, a document δ, and an indexed family of binary strings $(\tau_k)_{k\in\Gamma}$, with some $\Gamma \subset [\![0, w-1]\!]$ such that $|\Gamma| \le \kappa$. We define the validator predicate $\epsilon(\delta, \tau, S)$ to be true iff for all $k \in [\![0, d-1]\!]$, $\beta(T'_k, T_k)$ exists and

$$\omega(\|_{j=0}^{d} D'_j \parallel \delta, k) = \beta(T'_k, T_k),$$

where the indexed family $T' = T + \tau$ and the partial function β is the one defined in "Proof of stack security by reduction" section.

Proposition 4 *Given some $S = (D, T)$, δ and τ as per Definition 10, if $\epsilon(\delta, \tau, S)$ then $(D \triangleright \delta, T + \tau)$ is a valid depth-$(d + 1)$ stack over the same fabric.*

Proof follows from Definition 3 □

Winternitz stack protocol

The structures presented so far can be used to create a bipartite protocol where *neither* party can repudiate a transaction. Under public-key cryptography, the signer is unable to repudiate a properly signed document since the verifier holds the signer's authenticated public key and can prove to an adjudicator that whoever signed the document had to have knowledge of the signer's private key. The parties are assumed to be mutually adversarial to exclude collusion.[4]

Channel model We wish to minimise assumptions about the communication channel between the parties, bearing in mind that beneficial channel properties may depend on trust and/or shared confidential information. We require weak integrity, i.e. that a message sent by one party to the other and which fails the other party's validation test will be received intact after a finite maximum number of re-transmissions that does not depend on the message. No authentication of communicating parties is required; however countermeasures must be put in place to prevent an adversary from injecting messages in the channel at a rate that overwhelms the bona fide recipient. Because both parties are interested in progress, they can share a weak secret based on which all messages are extended with a MAC. However, even if no secret is shared, all protocols we present in the sequel require little computation at the receiving end for message validation (at most $\kappa + 1$ hash evaluations per message), so in practice each protocol contains its own DoS countermeasure, which may or may not be combined with other defences depending on the threat model. We mark received values with an asterisk $*$ to emphasise that they are not necessarily the same as those sent. Consider the following

Protocol 1 (Bipartite Winternitz Stack (BWS) protocol)

Parties: Alice(signer) and Bob(verifier)

Protocol parameter: fabric width $w \in \mathbb{N}$, w is a power of 2. **Initially:**

- **I1** In private: Alice chooses N, produces a random (w, N)-fabric F and saves it in local secure storage
- **I2** Alice publishes the fabric edge $E = E(F)$ and authenticates it out of band. E is now Alice's *public key*
- **I3** Alice invites Bob to participate in up to L transactions, $L \lesssim wN/\kappa$, see Eq. 3. The invitation and the value of L need not be authenticated
- **I4** In private: Bob creates a random length-L Winternitz chain $q^{[k]}$ and authenticates $Q = q^{[L-1]}$ out of band. Q is now Bob's public key, good for L transactions. Also Bob produces the initial stack $S_0 = (\emptyset, E)$ and saves it in local storage.
- **I5** Alice prepares document $\delta = L \parallel Q$ and computes a depth-1 stack over F: $S_1 = \mathbf{p}(Q, (\emptyset, E)) = (D_1, T_1)$,

[4] Note that no bipartite protocol can prevent collusion since the parties can destroy any documents and signatures by mutual consent and run the protocol from the beginning.

stores it, then communicates $\tau = T_1 - E$ to Bob.

I6 Bob receives τ^*, prepares the same δ and checks $\epsilon(D_1, \tau^*, S_0)$. If true, $\tau^* = \tau$ and Bob sends $q^{[L-2]}$ back to Alice and saves[5] $S_1 = (\emptyset \triangleright D_1, E + \tau)$ in local storage. If false, Bob requests retransmission of τ and repeats this step.

(Any non-receipt of a protocol message so far can be overcome by Automatic Repeat Query (ARQ) safely. If ARQ fails, this constitutes denial of service by the non-responding party or a protocol violation by the sender. Either way, the protocol fails.)

Repeat for $j = 2..L-1$:

R1 When Alice wishes to sign the next document δ_j, she computes

$$S_j = \mathbf{p}(\delta_j, S_{j-1}) = (D_j, T_j),$$

stores S_j and sends to Bob δ_j and $\tau_j = T_j - T_{j-1}$.

R2 Bob receives (δ^*,τ^*). Bob checks if he received and validated δ^* from the previous round $j-1$, and if so, (re-)sends $q^{[L-j-2]}$ to Alice[6] and remains in step **R2** of the current round. Otherwise, $\delta^* = \delta_j^*$ is fresh for the current round. Bob computes $\epsilon(\delta_j^*, \tau_j^*, S_{j-1})$. **If true**, $\delta_j^* = \delta_j$ and Bob stores[7] $S_j = (D_{j-1} \triangleright \delta_j, T_{j-1} + \tau_j)$, where $(D_{j-1}, T_{j-1}) = S_{j-1}$, and sends $q^{[L-j-1]}$ to Alice as the acknowledgement. The round is finished. **If false**, Bob sends a NAK and ignores δ_j^* and T_j^* as if they had not been received.[8]

R3 When Alice receives q^*, which could be a valid preimage or a NAK, she checks the truth value of $H(q^*) = q^{[L-j]}$. If true, she stores[9] $q^{[L-j-1]} = q^*$; the round is finished. **If false**, she sends (δ_j, τ_j) again and remains in step **R3**.

It is easy to see that the protocol is robust. Alice could only be one step behind Bob if she has not received Bob's acknowledgement in round j, since Bob goes straight to round $j+1$ after he sends it. But then Alice will re-send her message, for which the acknowledgement is missing, and Bob will be able to see that the message is from the previous round and will re-send his acknowledgement. According to the channel model stated earlier after a finite number of retransmissions Alice will receive the correct acknowledgement and the parties will synchronise. We assume that if a round has exceeded the maximum number of retransmissions, the protocol fails due to DoS.

Also note that step **R1** has the highest communication cost as κ hashes have to be communicated to Bob besides the document δ. The document is typically hash-sized, since the actual document text can be communicated out of band, with only its digest being signed by the protocol. It would not be efficient to send (δ_j, τ_j) again when, for example, only one element of τ_j is received with errors. However, there is an easy solution to this: send elements of τ_j one-by-one with immediate validation by Bob, who will hash them and compare the result with the stored stack top. This would limit retransmission to individual elements of τ_j. When κ of them have been received and confirmed, send δ_j.

Security of the BWS protocol

The security of the protocol rests on the following observations:

- Since the fabric is private to Alice, Lemma 1 applies, i.e. the combination of E and T uniquely defines D, making it impossible for Bob to forge Alice's signature for the current round. Hence Alice cannot repudiate a valid stack (D, T) over a fabric with the edge E, when it is claimed by Bob.
- However, Bob could claim to have received only some S_{j_B}, which is a substack S_j corresponding to a smaller depth $j_B < j$. For sufficiently small j_B Bob would be able to forge Alice's signature using fabric elements from S_j and thus repudiate the genuine one. However, this scenario is still impossible, since, according to step **R3**, Alice holds the acknowledgement $a = q^{[L-j-1]}$ at the end of round j and can prove that $\beta(a, q^{[L-1]}) = j$. So Bob's stack cannot have less depth than j, which is the same value as Alice's. Consequently Bob cannot repudiate.
- Alice could try to repudiate differently. Since the fabric is available to her in its entirety, she could build a different stack (D_j', T_j') of the same depth, not a substack of (D, T), with generally different documents D_j'. Alice could then claim that Bob has received and changed it. However, according to Lemma 1, Bob could only have done it if he had access to Alice's private fabric (and collusion is outside the threat model of any bipartite signature protocol). Consequently, any dispute between Alice and Bob regarding any

[5] Only the latest stack needs to be kept in storage as all the previous ones are its substacks.

[6] Here, as before, we assume that any document that Alice signs contains a nonce making it impossible for two documents to be the same.

[7] See footnote 5.

[8] The NAK (No AcKnowledgement) message can be implemented as no-reply, if Alice has a time-out mechanism at her end.

[9] Overwriting the previous value of q.

stack content should be resolved in favour of Bob automatically, provided that Bob's stack is valid.

Adjudication

BWS supports post-transaction adjudication by an algorithmic third party Judy, who need not authenticate Alice and Bob as long as she has the public values E and $q^{[L-1]}$ authenticated out of band. Keeping to the non-collusion scenario, Judy performs the following steps.

1. Judy requests from Alice the last $q = q^{[L-j-1]}$ she received from Bob, to determine the last j in the process of its validation using the public $q^{[L-1]}$. Alice will not benefit from reducing j to some $j_A < j$ since this would enable Bob to potentially forge her signature. However, Alice might chance it to falsely invalidate a few most recent rounds.
2. Judy requests Bob to provide the last q he sent, which should be the same $q^{[L-j-1]}$. If in the process of validation the value of j turns out to be some $j_B < j_A$, then Bob is lying and j_A is accepted as the value of d, since Alice has no access to Bob's chain and since the only source of valid q for her is Bob. If $j_B > j_A$, Judy accepts j_B as the correct value of d.
3. Judy requests a depth-d stack over a fabric with the edge E from Bob. Judy then validates the stack and confirms all signatures in it.

Practicalities

Now let us discuss how practical an implementation of the BWS protocol can be. The issue boils down to analysis of three major cost parameters: storage, computation/power and communication.

Storage

The issue of how much storage is required for the fabric is entangled with the issue of how much communication each round involves. Both are dependent on Eq. 4 and the chosen security parameter $\mathbf{Y} = \log_2 G$, which is plotted against the oracle parameter κ in Fig. 3. The curves are drawn up to a point at which $\kappa \log_2 w \simeq 512$. A further increase in κ would necessitate a longer hash value than the output of SHA-512 for oracle emulation, which may be a problem. Also one has to remember that Eq. 4 is only an approximation accurate for the region $\kappa \ll w$. For all the curves in Fig. 3 κ is at least one order of magnitude less than w, so the plots should be accurate enough.

If we limit the discussion to the case when the security parameter is at least 256, we can see that it is impossible to reach that level with the fabric narrower than 512 if the standard hash SHA-512 is to be used. The width 512 is sufficient for a lesser, 192-bit, security but in the quantum case it would be reduced to one half, 89 bits, which may not be sufficient. The wider the fabric, the less the critical value of κ. According to Eq. 3, the required storage capacity to store the whole fabric is

$$\lambda w N \simeq \lambda \kappa d_{\max},$$

where λ is the length of the hash used for the *chains* of the fabric (which is a parameter independent of the considerations of the oracle length). If the standard SHA-256 hash is used for the chains, $\lambda = 32$ bytes. For example, for a fabric sufficient for ~1 M signatures with security $\mathbf{Y} = 256$ it would appear that storage around 1 G bytes would be required ($\kappa = 31$, w = 4096), which is a large but completely feasible amount.

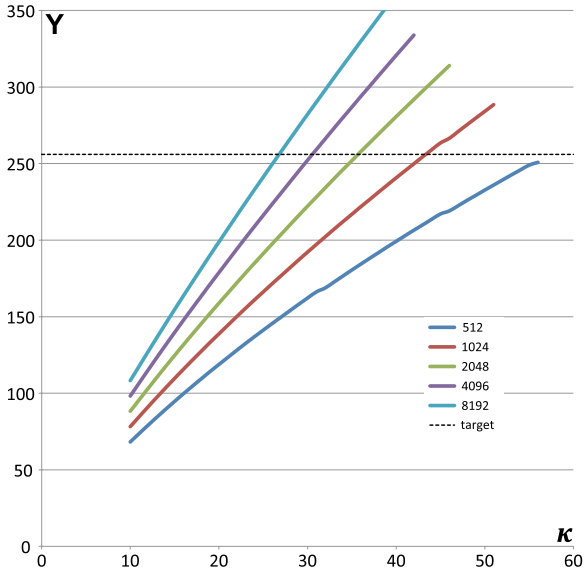

Fig. 3 Security parameter **Y** versus oracle size κ for various fabric widths. The dashed line marks the target security: 256 bits (or 128 bits Post Quantum). The target is reached at the following levels of w: **1024: 44, 2048: 36, 4096: 31, 8192: 27**

However, speed of access could be traded off for the storage requirement to reduce it by orders of magnitude using hash recalculation. Indeed, instead of storing every member of the fabric $r_i^{[k]}$, choose a large positive ϕ and substitute $\phi k_1 + k_2$, where $k_2 \in [\![0, \phi - 1]\!]$, for k. Only store elements $r_i^{[k_1,k_2]}$ when $k_2 = 0$. When in the round part of the protocol and a value of some $r_i^{[k_1,k_2]}$ is required for $k_2 \neq 0$, apply the hash function to $r_i^{[k_1,0]}$ k_2 times to obtain it.

It is worth mentioning that the modern GPU-equipped PC's hashrate (to say nothing about a cluster's) is measured in billions per second, which makes it possible to compute any required fabric element $r_i^{[k]}$ from the initial random $r_i^{[0]}$ in a matter of milliseconds

for any realistic fabric length, making it feasible to set $\phi = N$; only tens of kilobytes of the initial randoms would then be stored. However, the protocols presented in this paper, due to their very low computation cost, may be an attractive option for embedded systems and the IoT as well. A typical cost of a hash calculation on a microcontroller is 10 μs, down to 1 μs with hardware acceleration. Expanding a single node of the Winternitz chain $r_i^{[0]}$ to the next ϕ nodes for $\phi \sim 1000$ would be a matter of single milliseconds. It would only be required once in 1000 reads from the fabric, which would amortise the computational cost nicely, while reducing the storage requirements from gigabytes to the more affordable megabytes.

Note that the cost of recalculation is on average *one* hash calculation per fabric element, since elements are recalculated in bunches of ϕ every ϕ steps up the chain and stored in a temporary buffer. There is a difference between the average computation requirements, which affect *energy* consumption and the application execution *time* on the one hand, and the peak computation load, which affects *latency*, *power* requirements and *cooling* on the other, so the trade-off between storage and recalculation in any particular case might be more subtle.

All the above concerns Alice and Alice only. Bob has no access to the fabric. He only needs to store the documents and the top of the stack, which is the same size as Alice's public key E. The latter needn't be stored after the first document has been received, since on the one hand, E is not needed for the round part of the protocol, which deals exclusively with the top of the stack, and on the other, it can be reconstructed at any round j by popping documents off the stack in an obvious way (calculating the ω values of the documents starting from the last and working backwards). The need to produce E may arise if Judy is involved. However, storing E incurs only a small cost anyway, less than a 50% increase in the required storage capacity, and it is a diminishing fraction as more documents are received to be stored.

Communication The main consideration that drives the choice of the fabric width w is the length of the fabric edge. Since the edge is used as the public key identifying the signer, the latter is interested in having it as short as possible, thus increasing the required κ for a given security parameter. However, in step **R**1 of Protocol 1 Alice sends κ hashes (the difference between T_j and T_{j-1}) in addition to the document (which is typically represented by the digest of the document file and is one hash in length). Consequently, there is a trade-off between the public key length and the signature length. Figure 3 indicates that the variation of κ at our target level of security is rather limited, while the size of the public key doubles up every time we widen the fabric. On the other hand, the public key is only communicated once, in step **I**2, while step **R**1 of Protocol 1 is invoked as many times as there are documents to be signed before the fabric is used up. This points to the largest affordable w as the best solution. An increase in w also helps to reduce the fabric length N given the maximum depth d_{max}, which makes it possible to store fewer fabric elements for a given maximum recalculation cost. From this point of view, regimes close to $\kappa = 31$, $w = 4096$ seem optimal: 1 K bytes to send in step **R**1 as a document digest and its signature, and 128 K bytes to send in step **I**2 as a public key, both easily within the capabilities of a low-bit-rate communication facilities available to an IoT device.

Computations This is where the proposed protocol excels. Alice's costs for step **R**1 are trivial: one SHA-512 hash calculation as per Definition 6 and a few table lookups to fetch the fabric elements, if they are 100% stored. If they are recalculated, add κ SHA-256 calculations as recalculation cost. One might think that the cost of the SHA-512 will increase as j increases, since the new document is concatenated with all the previous ones thus making the hash argument ever longer, but this does not affect the cost. The mechanics of the hash algorithm are such that documents are processed block-by-block and the current state of the computation is used to produce the result. In round $j + 1$ the hash computation will simply proceed from the point that it reached in round j and will do the same fixed amount of work as that in round j. Another instance of hash calculation (SHA-256) is required at step **R**3 to validate Bob's acknowledgement.

Bob has a similar amount of work to do. At step **R**2, Bob must validate Alice's message, which will cost κ SHA-256 computations in any case, as it is not dependent on Alice's storage strategy; this is only a sub-millisecond time though, even for an IoT platform. There are also some table storage and retrieval operations to store the current top of the stack and retrieve the acknowledgement. If the acknowledgement chain is not 100% stored and requires recalculation, add the cost of another SHA-256, but that is all Bob is spending on computations.

MAWS protocol

Even though the BWS protocol is a two-party transaction, the verifier party (Bob) can only verify that the document has been signed by the signing party (Alice). If Bob disagrees with the document itself, the only option he has is to refuse to acknowledge it, in which case Alice can only repeat the step either indefinitely or until the protocol detects denial of service. The only way to continue would be to reinitialise the protocol with a new fabric at a significant cost.

In this section we will present a solution which gives Bob the power to (in)validate the document at the same time as signing for its receipt. Such a solution is available immediately with the BWS protocol if two stacks are used, one for either party, with Alice and Bob swapping roles for the second stack. This way Alice signs a document using her stack as the signer, and Bob acknowledges as the verifier, then Bob signs his acceptance of the document using his stack as the signer, and Alice acknowledges the receipt of the acceptance as the verifier.

However, it turns out that a single stack is sufficient to sign both the document and its acceptance. Under the bipartite protocol that we are about to present *both* parties are signers and both are verifiers, but one party has a significantly larger storage and communication (or, more precisely, transmission) requirements than the other.

Before we define the protocol, let us simplify the rules somewhat. Instead of stating it explicitly, we will now assume that each message is validated by the receiver and if the validation fails, a NAK is sent back to the sender, but no change of state occurs at the receiver as if the message were never sent. Also we assume that either the channel is authenticated, in which case the NAK is an authenticated message, or the channel only has weak integrity, and then the NAK is in fact a time-out of a duration exceeding the time required for the maximum number of retransmissions. In both cases the receiving party will be able to identify a NAK with certainty, but in the latter case the reaction to the NAK should be the same as the one to an invalid message, since those can always be injected in the channel by a DoS attacker over large enough period of time. Since we only require weak integrity, the protocols in the sequel will not differentiate between NAKs and invalid messages.

The protocol is fully asynchronous, i.e. each send requires a valid acknowledgement to be received. In the absence of an acknowledgement, the sending party resends its message up to the retransmission limit, then the protocol fails. The protocol does not require the channel between Alice and Bob to have absolute integrity; as before, we mark received values with an asterisk $*$ to emphasise that they are not necessarily the same as those sent.

Protocol 2 (Mutual Asymmetric Winternitz Stack (MAWS) Protocol)

Parties: Alice and Bob

Protocol parameter: fabric width $w \in \mathbb{N}$, w is a power of 2.

Initially: as in Protocol 1

Repeat for $j = 2, 4..L-1$ **[only even numbers]:**

R1 A new transactions document δ_j requires signing. Alice computes

$$S_j = \mathbf{p}(\delta_j, S_{j-1}) = (D_j, T_j),$$

stores it in local memory overwriting S_{j-1} and computes $\tau_j = T_j - T_{j-1}$.

R2 Alice sends (δ_j, τ_j) to Bob and goes to step **R**5 to await acknowledgement.

R3 Bob receives (δ_j^*, τ_j^*) and validates it by $\epsilon(\delta_j^*, \tau_j^*, S_{j-1})$. If valid, Bob concludes that

$$\delta_j = \delta_j^* \text{ and } \tau_j = \tau_j^*,$$

and stores

$$S_j = (D_{j-1} \triangleright \delta_j, T_{j-1} + \tau_j),$$

where $(D_{j-1}, T_{j-1}) = S_{j-1}$, in local memory overwriting S_{j-1}. If Bob approves δ_j, he forms his signature

$$\delta_j' = H\left(\delta_j \parallel q^{[L-j-2]}\right),$$

otherwise he sets δ_j' to zero.

R4 Bob sends the pair $(q^{[L-j-1]}, \delta_j')$ as the acknowledgement to Alice and goes to step **R**7 to await an acknowledgement.

R5 Alice retracts to step **R**2 unless she receives $(q^{[L-j-1]*}, \delta_j'^*)$ validated by $H(q^*) = q^{[L-j]}$. If continuing, Alice stores $q^{[L-j-1]} = q^{[L-j-1]*}$, computes

$$S_{j+1} = \mathbf{p}(\delta_j'^*, S_j) = (D_{j+1}, T_{j+1}),$$

and stores it in local memory overwriting S_j, while computing $\tau_{j+1} = T_{j+1} - T_j$.

R6 Alice sends the pair $(\delta_j'^*, \tau_{j+1})$ to Bob and waits for acknowledgement at step **R**9

R7 Bob retracts to step **R**4 unless he receives $(\delta_j'^{**}, \tau_{j+1}^*)$, validated by $\epsilon(\delta_j'^{**}, \tau_{j+1}, S_j)$. If valid, Bob concludes that

$$\delta'^{**} = \delta_j'^* \text{ and } \tau_{j+1}^* = \tau_{j+1}.$$

and stores

$$S_{j+1} = (D_j \triangleright \delta_j'^*, T_j + \tau_{j+1}),$$

where $(D_j, T_j) = S_j$, overwriting S_j,

R8 Bob sends $q^{[L-j-2]}$ as an acknowledgement to Alice. If $\delta_j'^* = \delta_j'$, the transactions is completed, and Alice knows it. Otherwise, Bob's signature δ_j' was

either zero or miscommunicated, and again, Alice knows it. The transaction is null and void; a (complete) repeat-round is necessary if both parties still wish to sign.

R9 Alice retracts to step **R**6 unless she receives $q^{[L-j-2]*}$ and validates it by $H(q^{[L-j-2]*}) = q^{[L-j-1]}$. If valid, she stores $q^{[L-j-2]} = q^{[L-j-2]*}$ and checks that

$$\delta_j'^{*} = H\left(\delta \parallel q^{[L-j-2]}\right).$$

If the equation holds, then the transaction is completed and Bob knows it. Otherwise Bob either rejected the transaction or his approval was miscommunicated, Bob knows which. Either way, the transaction is null and void; a (complete) repeat-round is necessary if both parties still wish to sign.

The protocol is generally robust as the sending of a message is paired with its validation and possible retransmission, except step **R**4, where Bob's signature is communicated but it cannot be validated *before* step **R**7 when Alice has already sent κ hashes and Bob's signature back. If the signature was corrupted in communication at step **R**7, this would waste the protocol round, in terms of both communication and fabric/chain material.

Security of the MAWS protocol The non-repudiation properties of MAWS hinge on the fact that Bob's signature is based on the pre-image of the latest member of Bob's Winternitz chain disclosed to Alice, namely $q^{[N-j-2]}$. Alice is unable to forge Bob's signature without knowledge of $q^{[N-j-2]}$, and when that value is disclosed to her in step **R**8, Bob's signature or refusal to sign has been signed by Alice already, in steps **R**5 and **R**6.

So it looks as though without hosting a Winternitz fabric, Bob can sign Alice's documents using nothing more than a single Winternitz chain. The post-transaction adjudication for MAWS is the same as that for BWS, see "Adjudication" section, except Judy also checks all δ' messages and marks the documents as approved or not approved accordingly.

Reverse Winternitz stack (RWS) protocol

Now let us tighten the communication model. Since the DoS defences would benefit from filtering incoming messages before the protocol calculations based on them are launched anyway, let us assume that Alice and Bob share a weak secret and use a symmetric Message Authentication Code (MAC, e.g. HMAC, based on the same hash function as the chains) to authenticate messages from Alice to Bob and back. Message authentication cannot be used to replace signatures since MACs are symmetric and can be repudiated. However, even a short MAC stops message insertion and message altering attacks very effectively. For example, a 32-bit MAC has less than one in a billion chance to be guessed in an attempt to insert or alter a message. On the other hand, for a known-plaintext attack to succeed in obtaining even a short AES128 key, the number of intercepted messages required is many orders of magnitude more than the length of any realistic Winternitz fabric.

Let us therefore adopt a more restrictive channel model whereby a message sent is extremely likely to be received correctly or not at all. Under such conditions MAWS becomes robust and any validation failure can safely be attributed to protocol violation by the sending party.

Given that, we are now able to propose a protocol where Alice has no independent signing function. All Alice does is certify Bob's signatures, effectively turning into a kind of secure signature server. Alice is unable to forge Bob's signature, nor Bob repudiate it. Assuming non-collusion, the mutually mistrustful Alice and Bob are still able to prove to Judy that Bob signed the documents he claims to have signed and to stop him repudiating his signature. The security of the following protocol trivially follows from the security of MAWS.

Protocol 3 (Reverse Winternitz Stack (RWS) Protocol)

Parties: Alice(signature server) and Bob(signer)

Protocol parameter: fabric width $w \in \mathbb{N}$, w is a power of 2.

Initially: as in Protocol 1

Repeat for $j = 2..L-1$:

R1 Bob computes the signature

$$s_j = H\left(\delta_j \parallel q^{[L-j-2]}\right),$$

where δ_j is the document he wishes to sign (or its digest, whichever is shorter), and sends it to Alice via an authenticated channel.

R2 Alice eventually receives s_j from Bob and computes

$$S_j = \mathbf{p}(s_j, S_{j-1}) = (D_j, T_j),$$

stores S_j and sends to Bob $\tau_j = T_j - T_{j-1}$.

R3 Bob eventually receives τ_j and validates it by $\epsilon(s_j, \tau_j, S_j - 1)$. If invalid, the protocol fails. Otherwise, Bob stores

$$S_j = (D_{j-1} \triangleright s_j, T_{j-1} + \tau_j),$$

where $(D_{j-1}, T_{j-1}) = S_{j-1}$, overwriting S_{j-1}. Bob also stores δ_j under the index j and sends q^{L-j-2} to Alice as the acknowledgement.

R4 Alice eventually receives $q^{[L-j-2]}$, checks that $H(q^{[L-j-2]}) = q^{[L-j-1]}$ and if so, stores $q^{[L-j-2]}$ overwriting $q^{[L-j-1]}$ and completes the round. Otherwise the protocol fails.

The protocol only fails if a party wilfully sends the wrong message. Failure to receive a response should be construed as a communication failure, not a security event, as the protocol stalls awaiting retransmission.

As before, a repudiation attempt from Bob on a given document δ will be countered by the retrieval of the latest known q from Alice and a stack of the corresponding depth from Bob. The verifier will then examine all s_j to find the one for which $s_j = H\left(\delta \parallel q^{[L-j-2]}\right)$, which proves the signature.

The last of the signed documents introduces an uncertainty as to whether or not Alice has completed step **R**4 on it or not, but it is not a major problem. The verifier may simply delay verification until Alice is quiescent.

Application of RWS to Internet of Things

The variety of IoT devices is very broad. It stretches from systems that have computation and communication capabilities approaching those of ordinary computers, to microcontroller-based smart sensors on a tight energy budget with low-bit-rate long-range (LoRa) radio communications. It is the latter category that presents unique challenges in network security, especially when nonrepudiation is required in a multi-vendor safety-critical system.

It is little appreciated in literature that IoT communication requirements are quite asymmetric. The success of sending data over the radio depends very much on the transmit power, which has to come out of the overall power budget of the device. IoT platforms tend to transmit little and do it infrequently. There are also legal constraints on the duty cycle and radiated power when operating in the frequency bands available to LoRa transmissions. However, receiving data is possible at a higher data rate spending much less power. In fact the power is used mostly for digital signal processing of the received signals, not the reception process as such; it can be reduced further by doing the processing less fast. In an IoT swarm, the edge server is typically equipped with a more powerful transmitter operating at a higher bit-rate. The IoT device can receive such a signal with less battery drain.

Ordinary non-repudiation protocols, e.g. those that involve cryptographically signing a transaction by both parties, exhibit symmetric computation and communication requirements. For example a 256-bit ECDSA produces a 512-bit or 64 byte signature, which is not quantum secure. Moreover, even using advanced microprocessors (rather than cheap microcontrollers), such as ARM Cortex-M4, the signature computation time is measured in hundreds of milliseconds (Fujii and Aranha 2019) compared to hundreds of microseconds for ~30 hashes that Bob computes in each round of RWS ($w = 4096$). The volume of data transmitted by Bob in one round of RWS is the same as it is under ECDSA, namely 32 bytes for the signature s and 32 bytes for the acknowledgment q. The volume of data to receive for Bob is much higher, close to 1 Kbyte. Finally, Bob retains the audit trail of all his signatures with the assurance that all of them (possibly with the exception of the very last one) have been registered by Alice and hence usable in transactions.

The audit trail forces Alice to be honest, especially when there is a system penalty if a proof against Alice is submitted by Bob. If Alice knows that, and if there exists some Proof of Stake for Alice (not necessarily digital), she can be used as Bob's proxy, making it unnecessary for a third party to communicate with Bob for signature validation frequently, especially if delayed validation is compatible with the security model. e.g. in the airplane black box type of application.

Related work

The idea of a hash-based signature scheme is classic, due to Lamport (1979). The basic approach is to associate a pair of nonces with each digit (one for the value 0 and the other for the value 1) of a message digest and use their hashes as the public key. The signer reveals the nonce associated with the value of the corresponding digit to form the message signature, which is only effective for one message. The scheme involves communication of very large signatures. This was improved upon by Merkle (1982) and Winternitz. The former paper proposes to only sign the digits whose value is 1 and to include a checksum to counter bit omission. The latter proposal (Winternitz One-Time Signature, WOTS) is to segment the digest into chunks and use each chunk as an iteration

counter in repeatedly hashing the corresponding nonce, again with a checksum guarding against reduction of iteration counters. WOTS was first published as an idea outline in Merkle's conference paper (Merkle 1988), reference 6 of which is to Winternitz's private communication. Neither Merkel nor Winternitz proposed anything to mitigate the one-time nature of Lamport's signature protocol, and the improvements are only in the signature size, which is much shorter than Lamport's, but is still very long compared to public-key cryptography with similar security parameters. This line of research has been continued further; more recent work includes a WOTS+ (Hülsing 2017) scheme, which extends WOTS, and XMSS (Hülsing et al. 2018), which extends the original Merkle proposal.

Another line of research was sprung by the seminal paper on HORS (Reyzin and Reyzin 2002), a "Hash to Obtain Random Subset" proposal, which turned out to be very fruitful. The idea here is to hash the digest and partition the hash image into equal length binary integers, the values of which are gathered into an index set. The indices select the pre-images to be revealed to form the message signature. An attacker would have to find a different message whose digest produces a subset of the index set under the hash function to obtain a counterfeit signature. The authors of Reyzin and Reyzin (2002) demonstrate that this is computationally hard. The hardness remains significant even when the set of pre-images used has expanded after signing a few messages. This approach and its successors are often referred to as *few-time* signature schemes. There is an elaboration of HORS by Hülsing in Bernstein etal. (2015), where the ideas of WOTS+ and an improved version of HORS, HORST, are combined. For the security parameter value 256, which we use as the typical case, the message signature claimed in Bernstein et al. (2015) is 41 thousand bytes, see Table 1 in that paper. This does not compare favourably with about one thousand bytes in the MAWS signature in the typical case, and even less with 64 bytes in RWS (all cases, ignoring input volume). However, the upside of their approach is a short public key, circa 1 K bytes, whereas our method would require a public key measured in hundreds of kilobytes (128 K in the typical case). Nevertheless, the public key can be stored in an embedded system or obtained by its hash via an unprotected public network, so we do not see the size of the public key as an important parameter.

HORS-like signatures can be shortened further: a recent paper, Lee and Park (2021) claims a reduction by more than a third, but the ballpark cost of communicating a signature of this size is still more than an order of magnitude more expensive than any message of our protocols. Finally, we are aware that the known few-time signatures have striven to rid themselves of the protocol state as this is seen as undesirable in the general security setting, see (Bernstein etal. 2015). Our protocols are clearly not stateless; however, for the application domain they are intended for it can be an advantage, since not only the transactions but also their ordering is assured by the signature stack; neither can be repudiated. At the end of the spectrum opposite to IoT, where there is continuous communication of large-volume data, the few-time hash-based signature approach has just seen an improvement (Li etal. 2023).

Protocol versus signature scheme

Since literature on digital signatures is dominated by papers focusing on signature schemes, such as the ones quoted above, the reader may expect the present work to be of the same nature, which it is not. We provide some comments below to draw an appropriate dividing line.

The research community has established certain quality criteria for digital signatures. The central notion for this is the one of *signature scheme*, which goes back to Goldwasser et al. (1988), where it was first formulated. A signature scheme is a triplet of probabilistic polynomial-time algorithms (G, S, V): a generator G, a signer S and a verifier V, which are used to create a pair (sk, pk) of a secret key and a public key, produce a signature s for a message M using the secret key sk, and to verify, using the public key pk, that the message signature s is valid for the message M, respectively. The scheme is correct iff $V(pk, M, S(sk, M))$ is true with probability 1 for any message M and any pair (pk, sk) produced by G. Implicit in this definition is the context-free nature of the scheme. Given the pk anyone should be able to validate any message M based on its s without knowledge of any previous messages that were sent or received by the principals. The proof of a signature is thus self-contained.

By contrast a signature protocol separates assurances given to the principals engaging in it and any self-contained proofs a principal is able to submit to an adjudicator. This separation is very fruitful, especially in our chosen case of limited resources, because adjudication is only required in abnormal situations (e.g. catastrophic break-down, where parties disagree on the origin of the message that caused it). Under normal operating conditions security proofs are only required between the principals, and those may rely on the state of the protocol *in addition* to the signature of the message and any keys. If the protocol guarantees that every principal has a consistent view of the state and if that can be successfully adjudicated on the basis of each party's data in case of disagreement (such as the above-mentioned break-down), then the shared protocol state enables the parties to drastically reduce communication without reducing

overall security. In our case it is the Winternitz stack, whose security properties have been rigorously studied in "Proof of stack security by reduction" section, that embodies the protocol state. Under a Winternitz stack protocol *all* previous messages influence the signature of the current one.

A Winternitz stack protocol can be treated as a digital signature scheme (G, S, V) only at the point of adjudication. Limiting ourselves to the last protocol, RWS, we could consider as S the mapping of all messages processed so far on the content of Alice's stack and Bob's response chain, which collectively represent the totality of all signatures. Then we could use as V the validation algorithm described in "Adjudication" section. However, strictly speaking this would not be a conventional signature scheme, since it does not apply to individual messages in isolation.

Finally, digital signature schemes are often required to resist adaptive attacks, a criterion which is known as Existential UnForgeability against an adaptive Chosen Message Attack (EUFCMA, or, more commonly, EUCMA, Jackson etal. 2019). The criterion is examined by assuming that the attacker can choose and submit messages for the signature oracle S polynomially often, with the messages produced one-by-one, taking into account the oracle's previous responses (adaptation). Since our constructed "signature scheme" does not apply to individual messages but rather to the totality of all messages received so far, adaptation would require re-running the protocol from the start, which is something no external attacker can do.

Limiting ourselves to RWS, the only possible adaptive actor is Bob, who himself is a party to the protocol. Bob is able to select message for signing and he must know all previous responses from Alice to sign another message, so the EUCMA question is valid even though the EUCMA setting is not. Adaptation would indeed be a concern for a protocol related to a few-time signature scheme, such as HORS (Reyzin and Reyzin 2002), where the mutual information between the message digest and the private key is less than the information contained in the digest. In the scheme presented in Reyzin and Reyzin (2002) a *set* is extracted by splitting the digest into fixed-size chunks. Theoretically, an attacker can choose the message to have a digest with many identical chunk-values thus reducing the size of the set, and with it the number of hash pre-images involved in the signature construction, which directly affects the signature security. In the present work the digest is turned into a *multiset* of a fixed number of elements. Consequently the choice of a message for an attack only influences which chains of a given fabric are used to reveal pre-images, not how many pre-images are revealed. The EUCMA property is thus assured by construction, requiring no further examination.

Conclusions

The Winternitz stack over a fabric has been proposed as a basis of post-quantum digital signature protocols. The security properties of the stack have been studied and a few protocols derived from them. The most interesting protocol, RWS, has a short signature size, 64 bytes, yet it is exclusively hash-based and ensures nonrepudiation. The security level of the protocol depends solely on Alice's available communication resources and Bob's storage constraints, but not on the parties' computational resources. For a reasonable amount of storage and compute power the protocol achieves 256-bit classical security or 128-bit quantum one, which does not diminish as more messages are signed without refreshing the public key. Under realistic assumptions at least 1 mln signatures can be made under the same public key, more if storage/recomputation is not a problem.

The protocol RWS places communication requirements on Alice and Bob asymmetrically, with Alice mostly transmitting and Bob receiving, which helps with the constraints of the low bit-rate communication characteristic of the IoT (in particular sensor networks). Alice can then act as a notary and Bob as her client without requiring any trust between them. The protocol provides sufficient assurances to Alice that Bob's signature is genuine, but if this needs to be proven to a third-party adjudicator *post hoc*, Alice and Bob must each supply their validation data. Given a no-collusion threat model, which is mandatory for all bipartite protocols, Bob cannot repudiate. Furthermore Bob cannot sign a document without making Alice aware of it. All of the above aspects closely match a typical IoT scenario.

Acknowledgements
Discussions with Bruce Christianson and his feedback are gratefully acknowledged.

Author Contributions
The single author contributed 100%.

Funding
No funding acknowledged at this time.

Availability of data and materials
None produced, none available.

Declarations

Competing interests
The authors declare no competing interests.

Received: 28 February 2023 Accepted: 26 February 2024
Published online: 04 April 2024

References

Bellare M, Rogaway P (1993) Random oracles are practical: a paradigm for designing efficient protocols. In: Proceedings of the 1st ACM conference on computer and communications security, CCS '93. Association for Computing Machinery, New York, pp 62–73

Bernstein DJ, Hopwood D, Hülsing A, Lange T, Niederhagen R, Papachristodoulou L, Schneider M, Schwabe P, Wilcox-O'Hearn Z (2015) SPHINCS: practical stateless hash-based signatures. In: Oswald E, Fischlin N (eds) Advances in cryptology—EUROCRYPT 2015. Springer, Berlin, pp 368–397

Fujii H, Aranha DF (2019) Curve25519 for the Cortex-M4 and beyond. In: Progress in cryptology—LATINCRYPT 2017. Springer, Berlin, pp 109–127

Goldwasser S, Micali S, Rivest RL (1988) A digital signature scheme secure against adaptive chosen-message attacks. SIAM J Comput 17(2):281–308

Hülsing A (2017) WOTS+—shorter signatures for hash-based signature schemes. Cryptology ePrint Archive, Report 2017/965

Hülsing A, Butin D, Gazdag S-L, Rijneveld J, Mohaisen A (2018) XMSS: eXtended Merkle signature scheme. RFC 8391:1–74

Jackson D, Cremers C, Cohn-Gordon K, Sasse R (2019) Seems legit: automated analysis of subtle attacks on protocols that use signatures. In: Proceedings of the 2019 ACM SIGSAC conference on computer and communications security, CCS '19. Association for Computing Machinery, New York, pp 2165–2180

Lamport L (1979) Constructing digital signatures from a one-way function, vol 238. Technical Report CSL-98. Technical report, SRI International

Lee J, Park Y (2021) HORSIC+: an efficient post-quantum few-time signature scheme. Appl Sci 11(16):7350

Li L, Lu X, Wang K (2023) eBiBa: a post-quantum hash-based signature with small signature size in the continuous communication of large-scale data. Comput J. bxad068

Merkle RC (1982) Secrecy, authentication and public-key cryptosystems. UMI Research Press

Merkle RC (1988) A digital signature based on a conventional encryption function. In: Pomerance C (ed) Advances in cryptology—CRYPTO '87. Springer, Berlin, pp 369–378

Reyzin L, Reyzin N (2002) Better than BiBa: short one-time signatures with fast signing and verifying. In: Batten L, Seberry J (eds) Information security and privacy. Springer, Berlin, pp 144–153

Rogaway P, Shrimpton T (2004) Cryptographic hash-function basics: definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In: Roy R, Meier W (eds) Fast software encryption. Springer, Berlin, pp 371–388

Publisher's Note