Signer-Optimal Multiple-Time Post-Quantum Hash-Based Signature for Heterogeneous IoT Systems

Kiarash Sedghighadikolaei^{a,*}, Attila A. Yavuz^a, Saif E. Nouma^a

^aDepartment of Computer Science and Engineering University of South Florida, Tampa, FL, USA

Abstract

Heterogeneous Internet of Things (IoTs) harboring resource-limited devices like wearable sensors are essential for next-generation networks. Ensuring the authentication and integrity of security-sensitive telemetry in these applications is vital. Digital signatures provide scalable authentication with non-repudiation and public verifiability, making them essential tools for IoTs. However, emerging quantum computers necessitate post-quantum (PQ) secure solutions, yet existing NIST-PQC standards are costlier than their conventional counterparts and unsuitable for resource-limited IoTs. There is a significant need for lightweight PQ-secure digital signatures that respect the resource constraints of low-end IoTs.

We propose a new multiple-time hash-based signature called *Maximum Utilization Multiple HORS* (MUM-HORS) that offers PQ security, short signatures, fast signing, and high key utilization for an extended lifespan. MUM-HORS addresses the inefficiency and key loss issues of HORS in offline/online settings by introducing compact key management data structures and optimized resistance to weak-message attacks. We tested MUM-HORS on two embedded platforms (ARM Cortex A-72 and 8-bit AVR ATmega2560) and commodity hardware. Our experiments confirm up to $40\times$ better utilization with the same signing capacity (2^{20} messages, 128-bit security) compared to multiple-time HORS while achieving $2\times$ and $156-2463\times$ faster signing than conventional-secure and NIST PQ-secure schemes, respectively, on an ARM Cortex. These features make MUM-HORS ideal multiple-time PQ-secure signature for heterogeneous IoTs.

^{*}Corresponding author

Email addresses: kiarashs@usf.edu (Kiarash Sedghighadikolaei), attilaayavuz@usf.edu (Attila A. Yavuz), saifeddinenouma@usf.edu (Saif E. Nouma)

Keywords: Lightweight cryptography; post-quantum security; Internet of Things (IoT); digital signatures; data structures.

1. Introduction

Authentication and integrity are essential for safeguarding sensitive data in next-generation networked systems, enabling the growth of the Internet of Things (IoT) across sectors like healthcare [1], military [63], and industry [72]. In healthcare, wearable devices transmit critical medical data, such as heartbeats and blood sugar levels, where compromised information can adversely affect individual health and security, underscoring the necessity for robust data protection measures.

Digital signatures provide scalable authentication with non-repudiation and public verifiability and, therefore, offer a trustworthy authentication alternative for embedded medical settings [60, 22, 68, 92]. However, traditional signatures like RSA [76] and ECDSA [47] are resource-intensive (due to operations such as modular exponentiation and elliptic curve scalar multiplication), causing performance degradation, leading to issues like frequent battery replacements in embedded medical devices [67, 5, 90]. There are lightweight conventional alternatives often employing a pre-computation approach [88] to reduce computation but in favor of increased storage. Hence, optimizing storage and computational efficiency is essential for extending device utility. Besides performance hurdles, emerging quantum computers can break these conventional-secure signatures and their aforementioned lightweight variants via Shor's algorithm [82]. Therefore, there is a critical need for PQ-secure digital signatures that can respect the resource limitations of low-end (embedded) IoT medical devices and applications.

NIST has standardized three classes of PQ-safe signatures [2, 26]: Lattice-based (\$\mathcal{LB}\$) CRYSTALS-Dilithium [29] and FALCON [34], and hash-based (\$\mathcal{HB}\$) SPHINCS+ [15]. Dilithium is based on Fiat-Shamir with Aborts [56] and offers a compact public key size but with larger signatures. Falcon relies on a hash-and-sign scheme using NTRU lattices [30] but necessitates more complex operations and floating-point arithmetic. Both are more computationally intensive, with large keys and signatures compared to their conventional counterparts, making them unsuitable for highly resource-limited devices. For instance, Dilithium and Falcon-512 are 10-77× slower than ECDSA and SchnorrQ on ARM Cortex A-72 and require substantial storage (117KB for Falcon-512, 113KB for Dilithium on ARM Cortex-M4) [49]. There are lightweight PQ-secure signatures that harness honest-but-curious verification servers with threshold control [9] and secure enclaves (e.g., Intel SGX [59]) for efficient verification and forward security [66].

However, these special architectural and security assumptions may hinder their adaptation to some IoT applications as well as introduce potential security risks (e.g., secure enclave vulnerabilities [85]).

 \mathcal{HB} standards like XMSS [19] and SPHINCS+ [15] provide strong PQ security through hash functions with minimal assumptions. They build upon Few Time Signatures (FTSs) [75, 54, 18] that permit efficient signing but only for a few messages. Stateful schemes like $XMSS^{MT}$ rely on Merkle Hash Trees (MHT) to enable Multiple-Time Signatures (MTS) from FTS, while stateless schemes such as SPHINCS+ use hypertree structures. Despite their merits, the signer overhead of these schemes is even higher than that of their lattice-based PQ-secure counterparts. Instead of general-purpose tree-based approaches, an alternative MTS is a conventional-secure public-key-based online/offline model. In this approach, public keys are pre-computed and stored at the verifier, while a hash-chain strategy is used at the signer with near-optimal efficiency [89, 93]. However, a straightforward transformation of FTSs such as HORS signatures [75] into MTS via such an online/offline approach is shown to be inefficient [89]. For instance, a HORS configuration with relatively short signatures leads to a key discard rate as high as 98%, even without considering weak message security [6]. These high loss rates are detrimental to the life span and practicality of the target application. In Section 7, we provide a more comprehensive revision of various alternative signatures and their pros and cons for low-end heterogeneous IoT applications. Overall, the state-of-the-art analysis suggests that there is a vital need for lightweight HB signatures that permit multiple-time signature generation capability but with significantly better signer performance than existing PQ-secure alternatives.

1.1. Our Contribution

We created a new multiple-time hash-based signature referred to *Maximum Utilization Multiple HORS* (MUM-HORS), which is specifically signed for heterogeneous IoT applications with embedded/wearable medical devices as a representative use case. MUM-HORS achieves PQ security with fast signing and compact signatures while permitting maximum public key utilization and key message security. Central to our scheme is a storage-efficient, fixed-size 2D bitmap for key management that facilitates efficient key derivation, public key management, signature failures, and weak-message security. We further outline the desirable features of our scheme as follows:

• Fast Signing and Efficient Weak Message Resiliency: The signing process of MUM-HORS is similar to efficient HORS, offering fast signing, short signatures,

and lower energy consumption. Additionally, we present an optimized mitigation method for weak message attacks [6] using XOR operations, which is significantly more efficient than techniques in HORSIC [54], HORSIC+ [55], and PORS [7]. We evaluated the signing efficiency (cycles and energy consumption) of MUM-HORS on various devices, including two embedded platforms (ARM Cortex A-72 and 8-bit AVR ATmega2560) and a commodity device. For instance, on the ARM Cortex A-72 (Table 1), MUM-HORS is faster by 24× than Falcon-512, by 200× than Dilithium, by 3900× than SPHINCS+, by 243-670× than XMSS variants, and even by 1.5-2× than conventional-secure schemes like ECDSA, Ed25519, and SchnorrQ.

- <u>Compact Key Management Data Structures</u>: Our key management data structure remains a small-constant size while offering a high capacity of signature generation. For instance, the size of our data structure is bounded as low as 1.4KB for a signing capacity of 2²⁰ signatures with 128-bit security, matching state-of-the-art multiple-time schemes like XMSS [20] and [89, 93]. We provide a comprehensive theoretical and numerical analysis of the stability and accuracy of our data structures, ensuring they remain compact and practical even for highly constrained devices (e.g., 8-bit AVR ATmega2560).
 - Full-fledge Implementation: MUM-HORS full implementation is available at:

https://github.com/kiarashsedghigh/mumhors

Potential Use Cases - Prioritizing Signer Optimality for Heterogeneous IoT Settings: MUM-HORS is a multiple-time signature designed for delay-tolerant and heterogeneous IoT applications that prioritizes signer efficiency and security (see Section 3). A typical application considered in state-of-the-art multiple-time signatures (e.g., [89, 93, 66]) is medical wearables (e.g., [33, 4]) and sensory devices in digital twins [3]. In these use cases, the battery-powered IoT device regularly takes measurements, digitally signs them, and periodically uploads them to a resourceful cloud server. Ensuring battery longevity, minimal cryptographic resource usage and long-term PQ security of the low-end device is of utmost importance. To enable this, MUM-HORS, as in conventional-secure multiple-time alternatives [89, 93, 66], relies on an offline/online public-strategy that requires pre-computed public keys to be stored in a resource verifier (e.g., cloud server). As discussed further in Section 7, this heterogeneous public-key model deviates from general-purpose signatures that rely on MHT or hypertrees (e.g., XMSS, SPHINCS) to avoid extreme burden on the signer in exchange for more storage on the cloud server. This is shown to be a highly favorable trade-off since

a cloud server can maintain a larger public key to enable the optimized security and longevity of low-end devices in consideration, an aspect of heterogeneous computing in IoTs.

Table 1: Comparison of Signature Schemes on ARM Cortex A-72 ($\kappa = 128$ -bit security)

Scheme	Signature G	eneration	Private	Signature	PQ	Sampling	Deterministic				
Scheme	8-bit AVR ATmega (cycles)	ARM Cortex A-72 (μs)	key (KB)	Size (KB)	Promise	Operations	Signing				
Full-time Signatures $(M=2^k)$											
ECDSA [47]	79 185 664	249.021	0.031	0.046	Х	Х	Х				
Ed25519 [48]	22 688 583	212.176	0.031	0.062	Х	Х	✓				
SchnorrQ [23]	3 740 000	196.395	0.031	0.062	Х	Х	✓				
Falcon-512 [34]	N/A	2 047.791	1.25	0.67	✓	1	Х				
Dilithium-II [32]	N/A	16 522.331	2.46	2.36	/	/	Х				
SPHINCS+ [15]	N/A	320 196.960	0.062	16.67	√	Х	✓				
M-time Signatures ($M=2^{20}$)											
SEMECS [89]	195 776	5.83	0.031	0.031	Х	Х	✓				
HORS [75]	342 976	46.8	0.031	0.78	√	Х	✓				
*HORSE [62]	342 976	46.69	790 MB	790 MB 0.078		×	,				
HORSE [02]	4 979 776	682.52	480	0.078	1	^					
**MSS [77]	5 792 000	N/A	1.438	2.295	√	Х	✓				
XMSS [19]	N/A	20 943.975	1.34	2.44	✓	Х	✓				
$XMSS^{MT}$ [42]	N/A	55 099.507	5.86	4.85	/	Х	✓				
MUM-HORS	637 376	129.32	1.43	0.78	/	Х	/				

The parameter setting is as in Table 2. To the best of our knowledge, there is no reported benchmark on an 8-bit microcontroller for XMSS variants, Falcon-512, Dilithium-II, and SPHINCS+. *It is impractical to implement HORSE variants on the AVR ATmega2560 MCU due to their large private key sizes. **The maximum MSS signing capability is 2^{16} since the EEPROM memory endurance is less than 2^{20} write/erase cycles.

2. Notations and Preliminaries

Notations: || and |x| denote concatenation and the bit length of x, respectively. $x \overset{\$}{\leftarrow} \mathcal{S}$ means x is chosen uniformly at random from the set \mathcal{S} . $m \in \{0,1\}^*$ is a finite-length binary message. $\{q_i\}_{i=a}^b$ denotes $\{q_a,q_{a+1},\ldots,q_b\}$. $\log x$ is $\log_2 x$. [1,n] denotes all integers from 1 to n. $f:\{0,1\}^* \rightarrow \{0,1\}^L$ and $H:\{0,1\}^* \rightarrow \{0,1\}^L$ denote one-way and cryptographic hash functions, respectively. a*b and a*b mean a=a*b and $a=(a*b\mod q)$, where * is an arbitrary operation.

Definition 1 A \mathcal{HB} digital signature SGN consists of three algorithms:

- $(sk, PK, I_{SGN}) \leftarrow \text{SGN.Kg}(1^{\kappa})$: Given the security parameter κ , it outputs the private key sk, the public key PK, and the system-wide parameters I_{SGN} .
- $\underline{\sigma} \leftarrow \text{SGN.Sig}(sk, m)$: Given the sk and message m, it returns a signature σ .
- $b \leftarrow \text{SGN.Ver}(PK, m, \sigma)$: Given PK, message m, and its corresponding signature σ , it returns a bit b, with b=1 meaning valid, and b=0 otherwise.

Definition 2 Hash to Obtain Random Subset HORS [75] is a \mathcal{HB} digital signature consists of three algorithms:

- $\underbrace{(sk, PK, I_{\text{HORS}})} \leftarrow \text{HORS.Kg}(1^{\kappa})$: Given the security parameter κ , it selects $\overline{I_{\text{HORS}}} \leftarrow (t, k, l)$, generates t random l-bit strings $\{s_i\}_{i=1}^t$, and computes $v_i \leftarrow f(s_i), \forall i=1,\ldots,t$. Finally, it sets $sk \leftarrow \{s_i\}_{i=1}^t$ and $PK \leftarrow \{v_i\}_{i=1}^t$.
- $\underline{\sigma} \leftarrow \text{HORS.Sig}(sk,m)$: Given sk and m, it computes $h \leftarrow H(m)$ and splits h into $k \log t$ -sized substrings $\{h_j\}_{j=1}^k$ and interprets them as integers $\{i_j\}_{j=1}^k$. It outputs $\sigma \leftarrow \{s_{i_j}\}_{j=1}^k$.
- $b \leftarrow \text{HORS.Ver}(PK, m, \sigma)$: Given PK, m, and σ , it computes $\{i_j\}_{j=1}^k$ as in $\overline{\text{HORS.Sig}(.)}$. If $v_{i_j} = f(\sigma_j)$, $\forall j = 1, \ldots, k$, it returns b = 1, otherwise b = 0.

Definition 3 Let $\mathcal{H} = \{H_{i,t,k,L}\}$ be a function family indexed by i, where $H_{i,t,k,L}$ maps an arbitrary length input to a L-bit subset of k elements from the set $\{0, 1, ..., t-1\}$. \mathcal{H} is r-subset (RSR) and second-preimage resistant (SPR), if, for every probabilistic polynomial-time (PPT) adversary \mathcal{A} running in time $\leq T$:

$$\begin{split} \operatorname{InSec}^{RSR}_{\mathcal{H}}(T) &= \max_{\mathcal{A}} \{ \Pr[(M_1, M_2, ..., M_{r+1}) \leftarrow \mathcal{A}(i, t, k) \\ & \text{s.t. } H_{i,t,k,L}(M_{r+1}) \subseteq \bigcup_{j=1}^r H_{i,t,k,L}(M_j)] \} < \operatorname{negl}(t, k) \\ & \operatorname{InSec}^{SPR}_{\mathcal{H}}(T) = \max_{\mathcal{A}} \{ \Pr[x \leftarrow \{0, 1\}^*; x' \leftarrow \mathcal{A}(x) \text{ s.t. } x \neq x' \\ & \text{and } H_{i,t,k,L}(x) = H_{i,t,k,L}(x') \} < \operatorname{negl}(L) \end{split}$$

3. Models and Use Cases

System Model and Use-case: We assume a traditional public-key-based authentication model for heterogeneous IoT-cloud applications, wherein low-end IoT devices gather information in a store-and-forward manner for a delay-tolerant setting. Embedded healthcare devices such as pacemakers and implantable devices are prominent examples [87, 78], in which the medical sensor takes continuous measurements (e.g., heart rate), digitally signs them, and then periodically uploads the telemetry and corresponding signatures into a cloud server for analysis. It is noted that other miscellaneous applications operate in similar settings, such as some smart cities [53] and drone services [79].

In our target use case with wearable medical devices, the longevity, security, and efficiency of the low-end device are of primary concern. Specifically, our digital signature scheme focuses on energy-efficient computation (battery life and

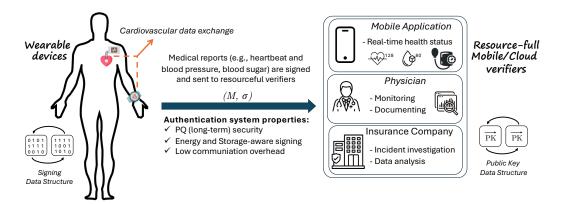


Figure 1: MUM-HORS system model for a resource-limited wearable Medical IoT use-case.

processing limits) and storage on resource-limited signers (e.g., 8-bit microcontrollers), while resourceful verifiers (e.g., cloud servers) manage public key storage and message authentication. Our system model is given in Figure 1.

<u>Threat and Security Model</u>: Our threat model assumes a quantum-computing capable adversary \mathcal{A} that can monitor all message-signature pairs and aims to intercept, modify, and forge them. The digital signature security model capturing our threat model follows the Existential Unforgeability under Chosen Message Attacks (EU-CMA).

Definition 4 The EU-CMA experiment for SGN is defined as follows:

- $(sk, PK, I_{\text{SGN}}) \leftarrow \text{SGN.Kg}(1^{\kappa}), (m^*, \sigma^*) \leftarrow \mathcal{A}^{\text{SGN.Sig}_{sk}(.)}(PK, I_{\text{SGN}})$
- \mathcal{A} wins the experiment after sending q queries if $1 \leftarrow \text{SGN.Ver}(PK, m^*, \sigma^*)$ and m^* was not queried to the signing oracle $\text{SGN.Sig}_{sk}(.)$.

$$\begin{aligned} Succ_{\text{SGN}}^{EU\text{-}CMA}(\mathcal{A}) &= Pr[Expt^{EU\text{-}CMA}(\mathcal{A}) = 1] \\ InSec_{\text{SGN}}^{EU\text{-}CMA}(T) &= \max_{\mathcal{A}} \{Succ_{\text{SGN}}^{EU\text{-}CMA}(\mathcal{A})\} < negl(T) \end{aligned}$$

4. Proposed Scheme

<u>Main Idea</u>: We propose MUM-HORS, a multiple-time hash-based signature scheme, which addresses severe key utilization limitations of HORS for multiple-time settings while ensuring maximum signer efficiency and weak message security. MUM-HORS follows public-key offline-online model, in which public keys are pre-computed from a cryptographic hash chain approach (offline) and maintained at the verifier side, while the signer generates a signature from a master key

efficiently (as in [89]). However, as discussed in Section 1 and shown in [89], this approach causes extremely large public keys and waste of individual components in public keys. This is due to the fact that, in HORS, to remain signer optimal by having efficient computation and communication, small values for k enable efficient signature generation and short signatures. However, following the offline model results in the waste of significant t-k keys. For instance, for the 128-bit security parameter setting (t=1024, k=25), only 2.44% of public keys are effectively utilized, which poses a huge degradation to authentication usability and verifier storage. Moreover, the HORS signature is vulnerable to weak message attacks [6] as during signing, it does not guarantee the derivation of k distinct index values (i_j), which causes the scheme to not meet the requirements of the r-subset-resiliency security (See Section 2).

To address the outlined challenges, MUM-HORS incorporates a storage-efficient two-dimensional bitmap with rt rows, each containing a t-bit list and metadata for correctness and optimization. During the signing, the first t set bits indicate available keys, with row and column indices aiding key derivation using the master key. Marked keys are used, while the remaining t-k keys stay unchanged, and depleted rows are replaced with new ones to maintain key availability. Moreover, MUM-HORS includes an efficient weak key mitigation strategy (consisting of XOR operations) through algorithm design and parameter setup.

Outline of MUM-HORS Algorithms: Algorithm 1 outlines MUM-HORS, and Algorithm 2 illustrates BM operations, as described below:

The MUM-HORS. Kg(.) initializes system parameters $I_{\text{MUM-HORS}}$, including the HORS and BM parameters (r,rt), representing the total and maximum bitmap rows, respectively. It generates the master key msk and three random pads (Step 1). An rt-row bitmap is created as a circular queue (with middle node deletion support) with rt cells, initializing global parameters head (first row's index), tail (last row's index), tail (last row's index), tail (last row's index), tail (unique (window size in bits), tail (last row number), activerows (current number of rows). Row-specific parameters include tail (current row number), tail (number of active bits in the row), and tail (the t-bit list) (Step 2). Public keys are generated by concatenating tail with each bit's row and column number (Step 3). The public key tail shares the bitmap parameters to ensure synchronization. The private key tail shares the bitmap parameters to ensure synchronization. The private key tail shares the bitmap parameters to ensure synchronization on the verifier (Step 4).

The MUM-HORS.Sig(.) first checks if the message hash produces distinct indices. If not, the hash is XORed with three pads; signing proceeds if any pad is effective (Steps 1-2). If none work, Ctr is concatenated with the hash and iteratively hashed until k distinct $\log t$ -sized parts are obtained (Steps 3-4). The hash

Algorithm 1 Maximum Utilization Multiple HORS (MUM-HORS)

 $(sk, PK, \text{BM}, I_{\text{MUM-HORS}}) \leftarrow \text{MUM-HORS.Kg}(1^{\kappa})$:

- 1: Set $I_{\text{MUM-HORS}} \leftarrow (I_{\text{HORS}}, r, rt), msk \stackrel{\$}{\leftarrow} \{0, 1\}^{\kappa}$, and $\{pad_i\}_{i=1}^3 \stackrel{\$}{\leftarrow} \{0, 1\}^{\kappa}$
- 2: Build bitmap $BM = \{row_i\}_{i=1}^{rt}$ and set $row_i.num = i$, $row_i.activebits = t$, and set all $row_i.bits[.]$ to 1. Set BM.head = 1, BM.tail = rt, BM.window = t, BM.nextrow = rt + 1, BM.activerows = rt, and $BM.activebits = rt \times t$
- 3: Create public keys $PK \leftarrow \{pk_i\}_{i=1}^r$ and set $pk_i.num = i, pk_i.activepks = t, \{pk_i.keys[j] \leftarrow f(PRF(msk||i||j))\}_{1 \leq i \leq r, 1 \leq j \leq t}$. Set $PK.head = 1, PK.tail = rt, PK.window = t, PK.nextrow = rt + 1, PK.activerows = rt, and PK.activepks = rt \times t$
- 4: Send the private key $sk \leftarrow msk$, $\{pad_i\}_{i=1}^3$, and BM to the signer, store the public keys PK and $\{pad_i\}_{i=1}^3$ on the verifier, and output the $I_{\texttt{MUM-HORS}}$

```
\sigma \leftarrow \texttt{MUM-HORS.Sig}(sk,m) \text{: Set } Ctr = 1 \text{ and } hash \leftarrow Trunc(H(m), k \log t)
```

- 1: Split hash into $k \log t$ -bit substrings $\{hash_j\}_{j=1}^k$ and interpret each as an integer i_j
- 2: if $\exists p,q \in [1,k]$ s.t. $i_p = i_q$ and $p \neq q$ then $hash \oplus pad_i$ and goto $1,i \in [1,3]$, else goto 5
- 3: Split $hash' \leftarrow Trunc(H(hash||Ctr), k \log t)$ as step 1 into integer indices i_j
- 4: **if** $\exists p, q \in [1, k]$ s.t. $i_p = i_q$ and $p \neq q$ then Ctr += 1 and **goto** 3
- 5: Compute $\{(r_j, c_j)\} \leftarrow \texttt{BITMAP_GET_ROW_COLUMN}(\texttt{BM}, i_j), \forall j \in [1, k]$
- 6: **return** $\sigma \leftarrow (\{PRF(sk||r_j||c_j)\}_{j=1}^k, Ctr)$

- 7: BITMAP_UNSET_INDICES(BM, i_i), $\forall j \in [1, k]$
- 8: if BITMAP_EXTEND_MATRIX(BM) == False then No more private keys to sign and exit

```
b \leftarrow \texttt{MUM-HORS.Ver}(PK, m, \sigma) : \mathbf{Set} \ hash \leftarrow Trunc(H(m), k \log t)
```

- 1: if hash or any $pad_{1,2,3}$ yield distinct i_j as steps 1-2 of MUM-HORS . Sig(.), then goto 3
- 2: Split $hash' \leftarrow Trunc(H(hash||Ctr), k \log t)$ as step 1 of MUM-HORS.Sig(.) into integer indices i_j . if $\exists p, q \in [1, k]$ s.t. $i_p = i_q$ and $p \neq q$ then Execute steps 3-4 of MUM-HORS.Sig(.) and return b = 0.
- 3: Retrieve public key pk_j for every index i_j similar to <code>BITMAP_GET_ROW_COLUMN(.)</code> and if $(pk_j \neq f(\sigma_j))$ return $b=0, \forall j \in [1,k]$. Otherwise, return b=1Verifier is Idle
- 4: Invalidate all the pk_i corresponding to the derived i_i as in BITMAP_UNSET_INDICES(.)
- 5: Extend the view of the public keys similar to BITMAP_EXTEND_MATRIX(.)

output is truncated to $k \log t$ bits for security (Steps 0 and 3). Once k distinct indices are identified, row and column indices of the first window set bits are retrieved from the BM using BITMAP_GET_ROW_COLUMN(.) (Step 5) to reconstruct the private keys as in key generation. The BITMAP_GET_ROW_COLUMN(.) (Algorithm 2) iterates over the bitmap rows to locate the $(index + 1)^{th}$ set bit and returns its row and column indices. The iteration follows the circular queue iteration algorithm. Finally, the signature is generated (Step 6).

When the signer is idle after sending a signature, it unsets the derived k in-

Algorithm 2 Bitmap Manipulation Algorithms

```
b \leftarrow \texttt{BITMAP\_EXTEND\_MATRIX(BM)}:
1: if BM.activebits < BM.window then
        if BM.nextrow \ge r then return False
3:
        if BITMAP\_CLEANUP(BM) == 0 then
4:
             Iterate over the rows and set index to the index of the row with the minimum .activebits
5:
              \label{eq:local_problem}  \mbox{Update } \mbox{BM}. active rows = 1, \\ \mbox{BM}. active bits = \\ \mbox{BM}[index]. active bits, \\ \mbox{and } \mbox{REMOVE\_ROW}(index) 
        Set fill capacity = min(rt - \texttt{BM}.activerows, r - \texttt{BM}.nextrow). Add fill capacity rows where for each, up-
     \text{date } \texttt{BM}.tail \overset{rt}{+=} 1 \text{ and set } \texttt{BM}[\texttt{BM}.tail].num = \texttt{BM}.nextrow, \\ \texttt{BM}[\texttt{BM}.tail].active bits = t, \\ \texttt{all } \texttt{BM}[\texttt{BM}.tail].bits[.] 
    to 1, and BM.nextrow += 1.
7: return True
    n \leftarrow \texttt{BITMAP\_CLEANUP(BM)}: Set cleaned = 0
1: Iterate over the rows and if row BM[i]. active bits == 0 then REMOVE\_ROW(i) and cleaned += 1
2: return cleaned
    (row, col) \leftarrow \texttt{BITMAP\_GET\_ROW\_COLUMN}(\texttt{BM}, index):
1: Iterate over the rows and if index \geq \text{BM}[i].active bits then index = \text{BM}[i].active bits 2: Set colidx as the index of (index + 1)^{th} set bit in \text{BM}[i].bits[.] and return (\text{BM}[i].num, colidx)
    BITMAP_UNSET_INDICES(BM, indices):
1: for index in indices do
        Iterate over the rows and if index \geq \mathtt{BM}[i].active bits then index = \mathtt{BM}[i].active bits
         Unset the (index + 1)^{th} bit in BM[i].bits[.] to 0 and update BM.activebits = 1 and BM[i].activebits = 1
    REMOVE_ROW(index):
1: if index == BM.head then BM.head = BM.head \stackrel{rt}{+=} 1
2: else if index == BM.tail then BM.tail \stackrel{rt}{--} 1
3: else
4:
         if {\tt BM}.head < {\tt BM}.tail then
             if index < \frac{\text{BM.} tail - \text{BM.} head}{2} then Shift rows from BM. head to index and set BM. head \stackrel{rt}{+}=1
5:
             else Shift from BM.tail to index and set BM.tail \stackrel{rt}{--}= 1
6:
7:
             if BM.head < index then Shift rows from BM.head to index and set BM.head \stackrel{rt}{+} 1 if index - BM.head <
8:
    rt-index o.w. Shift rows from BM.tail to index and set BM.tail \stackrel{rt}{-=} 1
             else Shift rows from BM.tail to index and set BM.tail \stackrel{rt}{-=} 1 if BM.tail - index < index o.w. Shift rows
    from BM.head to index and set BM.head \stackrel{rt}{\leftarrow} 1
```

dices in the bitmap using BITMAP_UNSET_INDICES(.) (Algorithm 2). Separating unsetting from index retrieval was a performance-driven decision. Private keys must correspond to the order of HORS indices derived from the message, requiring either a mapping data structure after sorting (required for unsetting) or separating these processes for efficiency. The indices are processed in descending order as computed in MUM-HORS.Sig(.) (Steps 1-4), locating each index's position in the bitmap, setting the corresponding bit to 0, and updating the bitmap's global and row parameters (Steps 1-3). After updating, the

signer checks if there are enough private keys for future messages by invoking BITMAP_EXTEND_MATRIX(.) (Algorithm 2) (Step 8). This function extends the matrix if fewer than window active bits are present. It first checks for and removes empty rows using BITMAP_CLEANUP(.) (Steps 2-3), which removes rows with zero active bits and returns the number of removed rows. If no rows are empty, the row with the fewest active bits is removed (Steps 4-6). Additional rows are then added to fill the BM, initialized, and the bitmap parameters are updated (Step 6). If no extension was needed or it was successful, it returns True (Step 7).

MUM-HORS.Ver(.) first verifies that the message hash and pads produce distinct indices (Step 1). If not, it checks the Ctr and, in the case of an invalid Ctr, the verifier rejects the signature and computes a valid Ctr to update the public key storage (Step 2). It then retrieves the public key corresponding to the i_j^{th} public key within the first window public keys. If any public key is rejected, the signature is deemed invalid; otherwise, the signature is accepted (Step 3). Finally, during idle periods, the verifier removes public keys from storage, similar to BITMAP_UNSET_INDICES(.), and if additional public keys are required, the verifier follows the BITMAP_EXTEND_MATRIX(.) procedure (Steps 4-5). To address potential resynchronization (a common challenge in MTSs [58]) between the signer and verifier caused by transmission noise or adversarial corruption, we propose a mitigation algorithm (see Appendix B) that helps with state recovery.

5. Performance Analysis and Comparison

This section analyzes the performance of MUM-HORS and its counterparts, focusing on signer storage overhead, key and signature sizes, and signing and verification times, with key generation occurring offline. We discuss the optimal selection of the row threshold (rt) in the bitmap and evaluate the computational complexity of each bitmap operation. Our analysis focuses on the implementation of BM as a circular queue with support for middle node deletion. An alternative linked-list version, detailed in Appendix A, enhances performance but requires additional memory for storing each node's successor address. We provide both analytical and detailed experimental analyses for the signer overhead on commodity hardware and two selected embedded devices.

5.1. Parameter Selection and Experimental Setup

The evaluation of MUM-HORS and its counterparts on various devices has been conducted with a security parameter set to $\kappa=128$ -bit, as detailed in Tables 1-3.

Hardware and Software Configurations: We used three types of devices for our evaluations. First, we used a desktop with an Intel i9-11900K @ 3.5GHz processor and 64GB of RAM to evaluate the signature generation and verification performance. Second, to demonstrate the signature generation performance of MUM-HORS for low-end (embedded) IoT settings, we used an 8-bit ATmega2560 @ 16MHz microcontroller with 256KB flash memory, 8KB SRAM, and 8KB EEPROM, as well as a Raspberry Pi 4 with a Quad-core Cortex-A72 @ 1.8GHz and 8GB of RAM. We only use a desktop to assess signature verification as our system model assumes a resourceful verifier. For the one-way and cryptographic hash functions (i.e., f and H), we choose Blake2¹ due to its high efficiency on commodity hardware and low-end embedded devices.

<u>Parameter Selection</u>: The counterparts' parameters are set according to their 128-bit security specifications. For MUM-HORS, the parameter relationship can be derived from the bitmap's two-dimensional structure, with M as the total number of messages to be signed and r as the total required number of rows:

$$M = (r - 1) \cdot \frac{t}{k} + 1$$

BM loads rt rows at a time, each containing t-bit vector, along with metadata (row number and the number of active bits). Therefore, the size of the bitmap is:

$$|BM| = rt \cdot (t + \log t + \log r)$$
 bits

The parameter rt affects the bitmap size and private key usage (as well as the number of signable messages). Increasing rt raises the chance of having rows with no active bits, as the remaining t-k bits are distributed across more rows before invoking <code>BITMAP_EXTEND_MATRIX(.)</code>. This reduces key loss by permitting the replacement of empty rows through <code>BITMAP_CLEANUP(.)</code> rather than removing the row with fewer active bits. For example, with parameters $(t=1024,k=25,l=256,r=25601,rt=11,M=2^{20})$, the private key usage is 100%, allowing all messages to be signed. However, setting rt=8 results in a loss of 10,536 bits and a reduction of 463 signable messages experimentally.

Optimal Values for rt: To derive the optimal bound for rt, we analyze the bitmap when the total number of bits is t, just before allocating a new row. The aim is to ensure that, with high probability, at least one row has fewer than k active bits

¹https://github.com/BLAKE2/

(ideally close to zero) so that key loss is minimized during row replacement when selecting k bits among the rows. We formally define the problem as follows:

<u>Question 1</u>: Given rt rows and t bits uniformly distributed among them, what is the minimum rt such that, with high probability, at least one row has a maximum load of k bits or less?

To answer this, we translate the problem into the Balls in Bins problem [74]. Specifically, given m balls and n bins, we aim to determine the maximum load in a bin with high probability after uniformly distributing the m balls. For this purpose, we apply Theorem 1 from [74]:

Theorem 1: Let M be the random variable that counts the maximum number of balls in any bin. We throw m balls independently and uniformly at random into n bins. Then with high probability, $M > load_{max}$ and we have:

$$load_{max} = \frac{m}{n} + \sqrt{2\frac{m\log n}{n}\left(1 - \frac{1}{\alpha}\frac{\log\log n}{2\log n}\right)}, \text{ if } m \gg n \cdot (\log n)^3, 0 < \alpha < 1$$
(1)

In the above context, m balls correspond to t bits, n bins correspond to rt BM rows, and α is the smoothing parameter. Increasing α provides a more conservative estimate of the maximum load, ensuring that, with high probability, the maximum load is almost the estimated bound. In essence, a higher α shifts the uniform ball distribution $\frac{m}{n}$ towards $load_{max}$. While this approach determines an upper bound on the maximum load, we are interested in having $\leq k$ bits in at least one row with high probability (minimizing the load). Thus, we define:

<u>Dual of Question 1</u>: Given a bitmap with $rt \cdot t$ bits, where t bits are present, and $rt \cdot t - t$ bits are marked as used, minimizing the number of unmarked bits in at least one row is equivalent to maximizing the number of marked bits in at least one row. What should rt be to ensure that, with $rt \cdot t - t$ marked bits added, the maximum load in at least one row is t - t or ideally t?

To solve this, we set $m = rt \cdot t - t$, n = rt and $load_{max} = t - k$ and we get:

$$t - k = \frac{rt \cdot t - t}{rt} + \sqrt{2 \frac{(rt \cdot t - t) \log rt}{rt} \left(1 - \frac{1}{\alpha} \frac{\log \log rt}{2 \log rt}\right)},$$
 (2)
if $rt \cdot t - t \gg rt \cdot (\log rt)^3, \quad 0 < \alpha < 1$

We analyze the parameters by implementing a numerical solver (in MATLAB,

available in our code repository²). With parameters t=1024 and k=25 and setting $\alpha=0.999$, our solver outputs a row threshold of 10.903, assuming a maximum load of t-k. To increase the likelihood of full depletion, assuming $load_{max}=t$, the solver outputs a safer margin of 13.94 rows.

5.2. Efficiency Evaluation and Comparison

Given the performance evaluation on two embedded platforms and a commodity device in Tables 1-3, takeaways are as follows:

Table 2: Comparison of Signature Schemes on Commodity Hardware ($\kappa = 128$ -bit security)

Scheme	Sig Gen	Sig Size	sk Size	Sig Ver	PK Size	PQ	Memory Exp	Sampling	Deterministic	
Scheme	(µs)	(KB)	(KB)	(μs)	(KB)	Promise	Code Size	Operations	Signing	
Full-time Signatures ($M=2^{\kappa}$)										
ECDSA [47]	15.81	0.046	0.031	46.24	0.062	Х	L	Х	Х	
Ed25519 [48]	12.14	0.062	0.031	33.17	0.031	Х	L	Х	✓	
SchnorrQ [23]	8.57	0.062	0.031	15.42	0.031	X	L	Х	1	
Falcon-512 [34]	170.45	0.67	1.25	28.21	0.87	✓	M	✓	Х	
Dilithium-II [32]	243.07	2.36	2.46	59.05	1.28	✓	M	✓	Х	
SPHINCS+ [15]	5217.01	16.67	0.062	415.41	0.031	✓	Н	Х	1	
M -time Signatures ($M=2^{20}$)										
Zaverucha et al. [94]	14.97	0.046	0.031	28.2	4.06GB	Х	L	Х	Х	
SEMECS [89]	0.68	0.031	0.031	11.9	96MB	X	L	X	✓	
HORS [75]	6.12	0.78	0.031	6.35	32GB	✓	L	Х	✓	
HODGE (63)	6.19	0.78	790 MB	6.41	32	,	Н	,	,	
HORSE [62]	86.25	0.76	480	6.46	32	•	п	_ ^	•	
XMSS [19]	2778.1	2.44	1.34	751.17	0.062	✓	Н	Х	✓	
$XMSS^{MT}$ [42]	6785.85	4.85	5.86	1297.94	0.062	✓	Н	Х	✓	
MUM-HORS	17.56	0.78	1.43	19.17	800MB	√	L	Х	✓	

The message size is 32 bytes. In the memory expansion and code size column, H denotes high, M denotes medium, and L denotes low. For SPHINCS+, parameters are (n,h,d,t,k,w)=(16,66,22,64,33,16). For XMSS, XMSS-SHA2_20_256 was selected, and for XMSS MT , XMSSMT-SHA2_20/2_256 was used. HORS and HORSE parameters are (t,k,l)=(1024,25,256), with d for HORSE set to 25290, where $d=M\cdot(1-e^{-k/t})$. HORSE can reduce memory by using more hash computations (Jakobsson [46] method), which, while decreasing private key size, increases signing time. HORS has been converted to an M-time signature using the same method as MUM-HORS. For MUM-HORS, parameters are (t,k,l,r,rt)=(1024,25,256,25601,11). The public key size for MUM-HORS is 25600 full rows plus one row with 25 keys, totaling 800.00076 MB. The f(.) that is used for HORS, HORSE, and MUM-HORS is Blake2-256.

Signer and Verifier Memory Usage: The MUM-HORS signer storage includes a 256-bit master key and a 1.4KB bitmap. The total storage overhead of 1.43KB is comparable to Falcon-512 and XMSS and is $1.7 \times$ smaller than Dilithium-II and $4 \times$ than XMSS^{MT}. Compared to HORSE, which uses hash chains, MUM-HORS offers 335-500,000× memory savings. MUM-HORS has low memory expansion and

 $^{^2 \}verb|https://github.com/kiarashsedghigh/mumhors/Optimizations/row_threshold.m|$

Table 3: Energy Usage of Signature Generation and Transmission on an AVR ATMega2560 MCU

Scheme	Signatu	re Generation	Signature	Transmission	Total Energy	Max Signing	Private	PQ				
Scheme	Time (cycles)	Energy Cost (mJ)	Sig Size (KB) Energy Cost (m		Cost (mJ)	Operations	key (KB)	Promise				
Full-time Signatures $(M=2^{\kappa})$												
ECDSA [47]	79 185 664	332.285	0.046	0.065	332.35	20 316	0.031	Х				
Ed25519 [48]	22 688 583	92.343	0.062	0.086	92.429	73 063	0.031	Х				
SchnorrQ [23]	3 740 000	15.222	0.062	0.086	15.308	440 946	0.031	Х				
M -time Signatures ($M = 2^{20}$)												
SEMECS [89]	195 776	0.797	0.031	0.043	0.84	8 035 714	0.031	✓				
HORS [75]	342 976	1.396	0.78	1.075	2.471	2 731 688	0.031	✓				
*MSS [77]	5 792 000	23.573	2.295	29.964	53.537	126 081	1.438	1				
MUM-HORS	637 376	2.594	0.78	1.075	3.669	1 839 738	1.43	/				

The parameter setting is as in Table 2.

code size compared to PQ standards. On an ARM Cortex-M4, Falcon-512 needs 117KB, Dilithium 113KB, and SPHINCS+ 9KB for signing and verifying [49].

MUM-HORS verifier storage is $5\times$ smaller than Zaverucha et al. [94] and $40\times$ smaller than HORS and deletes the public keys from the memory as they are used. Communication Bandwidth Overhead: Smaller k values in MUM-HORS reduce the signature size with fewer private keys. The signature size is comparable to Falcon-512, $3\times$ smaller than Dilithium-II, and $21\times$ smaller than SPHINCS+. MUM-HORS has similar signature size as HORS and HORSE, is $6.21\times$ smaller than XMSS MT , but is larger than non-PQ secure SEMECS, Zaverucha et al. [94], and full-time ECDSA, Ed25519, and SchnorrQ counterparts.

Signing on the Commodity Device and Verification: MUM-HORS achieves fast signing and verification. Signing is $17\times$, $24\times$, and $511\times$ faster than Falcon-512, Dilithium-II, and SPHINCS+, respectively, and $1.5\times$ and $1.2\times$ faster than ECDSA and EdDSA, respectively. While performance is comparable to ECDSA, Ed25519, and [94], MUM-HORS is $665\times$ faster than XMSS MT . Verification speed matches signing speed, making MUM-HORS $118\times$ and $38\times$ faster than PQ-secure XMSS MT and SPHINCS+, respectively, and $4.5\times$ and $3\times$ faster than ECDSA and EdDSA. Moreover, BITMAP_CLEANUP(.) takes $0.027\mu s$, BITMAP_UNSET_INDICES(.) $0.18\mu s$ per index, and BITMAP_GET_ROW_COLUMN(.) $0.23\mu s$ on commodity hardware, with the latter being the only one active during signature generation. It should be noted that BITMAP_EXTEND_MATRIX(.) does not always remove rows as it is conditioned on the presence of window bits in the BM.

Signature Generation on Embedded Devices: The efficiency of MUM-HORS on embedded devices (ARM Cortex A-72) has been presented in Table 1. On the ARM Cortex A-72, MUM-HORS signing is 1.75× slower than its baseline HORS due to bitmap operations required for achieving a 41-fold public key size reduc-

^{*}The maximum MSS signing capability is 2^{16} since the EEPROM memory endurance is less than 2^{20} write/erase cycles.

tion. NIST finalists, like Dilithium, show significant delays in signing (e.g., 16 seconds), making them unsuitable for real-time applications. Moreover, operations BITMAP_CLEANUP(.) takes $0.6\mu s$, BITMAP_UNSET_INDICES(.) takes $0.89\mu s$ per index, and BITMAP_GET_ROW_COLUMN(.) takes $1.53\mu s$.

Energy Impact and Cost of Signing on Embedded devices: In Table 3, we present the analysis of the impact of signature generation on energy consumption and battery life in 8-bit AVR microcontrollers using the energy estimation model from [71] based on the MICAz sensor node. The MICAz node features an ATmega128L microcontroller (16 MHz, 4KB EEPROM, 128KB flash memory) and a ZigBee 2.4 GHz radio chip (CC2420). It is powered by AA batteries with a maximum energy capacity of 6750J. According to [71], the ATmega128L MCU consumes 4.07nJ per cycle, while the CC2420 transceiver chip draws 0.168mJ per bit transmitted. Using these results, we estimated the energy consumption for signing and transmission for MUM-HORS and its counterparts.

The signing capability depends on both signature generation and transmission efficiency. Notably, MUM-HORS can support a larger number of signatures (2^{20}) than the practical limit of signing operations achievable on a MICAz node ($\approx 2 \cdot 10^6$). Our \mathcal{HB} counterpart, the Merkle Signature Scheme (MSS) [77], supports only $\frac{1}{14}$ of the signatures achievable with MUM-HORS. MSS uses Winternitz One-Time Signature (WOTS) and constructs a Merkle tree, as in XMSS [19], leading to higher signing costs due to the WOTS signature process and path computation in the tree. Our second \mathcal{HB} counterpart, HORS, offers $1.48\times$ more signatures than MUM-HORS, but at the cost of a significantly larger public key, approximately $40\times$ larger. The elliptic-curve-based SEMECS can generate more signatures than the MICAz node's practical limit. However, it does not provide post-quantum security. MUM-HORS offers the optimal balance between maximum signing capability, post-quantum security, and compact key sizes.

6. Security Analysis

Theorem 2 MUM-HORS is EU-CMA secure if H() is r-subset-resilient and second-preimage resistant:

$$\mathit{InSec}_{\mathit{MUM-HORS}}^{\mathit{EU-CMA}}(T) = \mathit{InSec}_{H}^{\mathit{RSR}}(T) + \mathit{InSec}_{H}^{\mathit{SPR}}(T) < \mathit{negl}(t,k,L)$$

Proof: Given a set of adaptively chosen and queried q valid message-signature pairs $\{(m_i, \sigma_i)\}_{i=1}^q$, there are the below cases where the adversary \mathcal{A} can forge a signature:

- $\frac{A \ breaks \ r\text{-subset-resilient of } H}{k}$: The A identifies a message m^* such that its k distinct elements (as in MUM-HORS.Sig(.)) are among the observed $q \cdot k$ elements from the last q messages, with a success probability of $(\frac{q \cdot k}{t})^k$. We note that our efficient mitigation method against weak message attacks ensures the derivation of k distinct indices from a message, either by using its initial hash XORed with random pads secured by long-term certificates on the verifier, or through an iterative procedure using an incremental counter Ctr.

The success probability $(\frac{q \cdot k}{t})^k$ decreases to $(\frac{k}{t})^k$ due to the bitmap design. Once the signer selects and uses k bits from the first window, they are marked as used, preventing reuse in future rounds. Uniqueness is maintained by the row parameter num, ensuring each bit's combination (msk||row||col) is distinct. Even when BITMAP_EXTEND_MATRIX(.) is invoked, new rows have unique row numbers, enforced by the global nextrow parameter. Moreover, the verifier maintains identical global and row parameters to manage public keys, ensuring synchronization with the signer. This guarantees that once a private key is used, the corresponding public key is invalidated during verification, preventing reuse.

In summary, since the bitmap replaces k out of t private keys, and the remaining t-k unused keys are independent and hidden from the attacker, the success probability per round is reduced to that of HORS as an OTS, which is negligible for suitable k and t. This probability of forging a signature on HORS, in addition to $(\frac{q \cdot k}{t})^k$, entails the likelihood of inverting the one-way function f(.) to derive private keys from the verifier's stored public keys. Grover's algorithm [37] can reverse a black-box function with input size N in $O(\sqrt{N})$ steps and $O(\log_2 N)$ qubits. For L-bit output f(.), the probability of reverting k public keys is $2^{-k \cdot \frac{L}{2}}$.

A breaks the second-preimage resistance of H: We evaluate the security of our hash function H(.) using Grover's model. The attacker could find m^* such that $H(m_i) = H(m^*)$ and produce a valid (m^*, σ_i) . Given identical hashes, the k indices for m^* will match those for m_i by any method (initial hash, pads, or Ctr). For an L=256-bit hash function like Blake2-256, the collision probability is $\frac{1}{2^{\frac{1}{2}}}$, providing 128-bit security, which is negligible. Moreover, the parameters k and t impact the security of H(.), requiring $k \log t = L$. If $k \log t < L$, the attacker's success probability increases to $\frac{1}{2^{\frac{k \log t}{2}}}$. To maintain security, we ensure $k \log t = L$ by truncating hash outputs to $k \log t$ bits using the Trunc(.) function during the signing.

Overall, we conclude with an upper bound:

$$\mathrm{InSec}_{\text{\tiny MUM-HORS}}^{EU\text{-}CMA}(T) < max(\frac{1}{2^{\frac{L}{2}}},\frac{1}{2^{\frac{k \cdot \log t}{2}}}) + 2^{-k \cdot \frac{L}{2}} + (\frac{k}{t})^k$$

We set the length of the master key (|msk|) and private key (HORS l parameter) to κ bits to ensure the minimum required security.

7. Related Work

Symmetric-key based Approaches: Two primary schemes, namely HMAC [10] and symmetric ciphers [83] are commonly used in IoT systems for their computational efficiency [38]. However, these approaches lack scalability in large, distributed systems and do not provide public verifiability, which is essential for health audits [39] and non-repudiation in legal contexts [21]. Digital signatures [47, 11, 76], though less efficient computationally, support scalable, publicly auditable authentication.

Conventional Digital Signatures: Conventional digital signatures like ECDSA [47], Ed25519 [11], RSA [76], and BLS [16] are considered for IoT [69], wireless spectrum management [36], 6G [50], and various other network domains. However, security protocols relying on these primitives have been shown to deplete battery life in resource-constrained devices and may require frequent undesirable maintenance [67, 5]. For example, while RSA is efficient in verification, it requires expensive signing operations and has large key sizes, and although ECDSA improves efficiency with smaller keys, it still incurs high computational costs. Moreover, relying on traditional intractability assumptions leaves these schemes vulnerable to quantum attacks like Shor's algorithm [82], with ECC-based ones being more susceptible than RSA [12, 40, 35, 73].

Lightweight Conventional Signatures with Special Trade-offs: Efforts to develop lightweight digital signatures that improve conventional methods like ECDSA and SchnorrQ often rely on advanced pre-computation and storage techniques, as done by SCRA [88] and Nouma et al. in [65]. These schemes rely on the online/offline signature paradigm [81] involving pre-computed tokens. For instance, Rapid Authentication (RA) [91] leverages pre-computed token aggregation for fast online signature generation. SEMECS [89] optimizes EC-based signature schemes by reducing signature and private key sizes of [90], addressing the computational burden of deriving ephemeral keys in Schnorr [80]-like signatures (e.g., ECDSA, SchnorrQ) through pre-computing them and storing their hash commitments on the verifier. Despite their merits, these schemes are also vulnerable to quantum computers as they rely on traditional intractability assumptions (e.g., (EC)DLP).

PQ-secure Signatures: The NIST-PQC standardization [25, 26] features \mathcal{LB} signatures CRYSTALS-Dilithium [32] and Falcon [34], alongside HB SPHINCS+ [15] for PQ-secure signatures. Dilithium, based on the Fiat-Shamir with Aborts principle [56] and M-LWE and M-SIS problems [17, 52], uses a uniform distribution to reduce the public key size by about half, although it results in slightly larger signatures and its variable signing time may impact performance on resourcelimited devices. Falcon, a hash-and-sign scheme [31] using NTRU lattices [30], requires double-precision floating-point arithmetic and complex operations like Fast Fourier Transform and matrix decomposition, making it unsuitable for devices lacking a Floating-point Unit (FPU). While for 128-bit security, Falcon's public key and signature sizes (897 and 690 Bytes, respectively) are smaller than Dilithium's (1.28 KB and 2.36 KB, respectively), both are less appropriate for resource-limited devices due to their complexity, larger key and signature sizes, and vulnerability to side-channel attacks [8, 24, 27]. Currently, there are no deployable open-source implementations of \mathcal{LB} digital signatures for highly resourcelimited embedded devices (e.g., with 8-bit microprocessors), aside from BLISS [28], which was not selected as NIST-PQC standard due to heavy side-channel attack vulnerabilities [84].

Lightweight PQ-secure Signatures with Special Assumptions: Recent lightweight PQ-secure signature schemes rely on specific architectural features and distributed computation to enhance the signer efficiency. ANT [9] transforms a \mathcal{LB} OTS into a polynomially-bounded many-time signature through distributed public key computation under the assumption of honest-but-curious servers and passive adversaries. HASES [64] and its extension [66] convert HORS signature [75] into a many-time signature, assuming a secure enclave (e.g., Intel SGX [59]) to store the private key, enabling public key derivation and forward security. Despite their signer efficiency, these special assumptions on the verifier side may limit the applicability of these solutions in some NextG IoT settings.

Hash-based Signatures and their Building Blocks: Unlike \mathcal{LB} schemes, \mathcal{HB} standards such as XMSS [41] and SPHINCS+ [15] rely on minimal intractability and number-theoretical assumptions, offering strong PQ security. These schemes use cryptographic hash functions like SHA-256, which are widely available, facilitating the transition to PQ secure options. While some \mathcal{HB} signatures provide efficient signing and verification for a limited number of signatures [75, 54, 18], others, such as [13, 44, 20], offer high security for extended usage but involve large signature sizes and costly signing processes.

HB FTSs, built on OTSs [51, 61, 18, 43], allow a few signature generations

with the same key pair, though security diminishes with each additional message. The first FTS, BiBa [70], prioritized fast verification and small signature sizes but had trade-offs in signing time and public key size. Subsequent FTSs, like HORS [75], built on Lamport OTS with Bos-Chaum signatures and cover-free families, and its variants [13, 55, 54, 62, 86, 92], and others like PORS [7] and FORS [15] enhanced the robustness against weak message attacks [6]. However, they have large private keys (e.g., HORSE [62]) and signing times (e.g., HORSIC+ [55]), or are time-valid secure (e.g., TV-HORS [86]).

Stateless \mathcal{HB} schemes like SPHINCS [14] and SPHINCS+ [15] use a hypertree structure to extend FTSs to full-time signatures. This structure involves Merkle trees with Winternitz OTS [18] as leaves, where each node signs 2^h child nodes, with h as the tree height. For example, with h = 50, a single key can generate one million signatures per second over 30 years. The hypertree's leaves are FTSs (e.g., HORST instances [14]), which allow multiple message signing, enhance path collision resilience, and reduce tree height. While this approach decreases signature size using fewer OTS instances, it increases signing time due to Merkle tree generation. Despite implementations for resource-limited devices (e.g., Armed-SPHINCS [45]), these schemes are computationally intensive, with SPHINCS+ being $330 \times$ and $21 \times$ slower than ECDSA and Dilithium, and its signatures being $362\times$ and $7\times$ larger, respectively. Moreover, stateful \mathcal{HB} signatures, such as XMSS MT [42] and LMS [57], provide strong security and features like forward security. They use a Merkle tree to group 2^h OTS key pairs into one signature key pair, with the root as the public key and each leaf as an OTS. A Merkle signature includes the OTS key pair index, the OTS signature, and the authentication path. However, their high computational cost and memory usage limit their practicality for resource-constrained devices. For example, $XMSS^{MT}$ signatures can be 4.85 KB, which is $105 \times$ and $2 \times$ larger than ECDSA and Dilithium signatures, respectively, and the signing is $430\times$ and $30\times$ respectively slower.

Among \mathcal{HB} signature schemes, HORS [75] is valued for its efficiency and underpins signatures like XMSS^{MT} [19] and SPHINCS+ [15]. However, extending HORS FTS to full-time signatures using hypertree-based (e.g., SPHINCS+) or tree-based (e.g., XMSS^{MT}) methods is inefficient for resource-constrained devices. An alternative is the online/offline paradigm (as in [89]), where public keys are pre-computed and stored offline, with one key used per signing round. While HORS benefits from small k values for short signatures, it requires large t for adequate security (e.g., 128-bit). For example, signing t messages with 128-bit security requires t = 1024 and t = 25, leading to 32GB of public key storage. In each signing round, 25 keys are used and 999 discarded, resulting in a 97%

key loss and only 2.44% effective utilization of the public key storage. This inefficiency affects device utility, authentication lifetime, and computation/storage requirements. To reduce key discards in HORS, private keys can be tracked via indices (one per key), with each t key derived from the master key padded with the corresponding index. However, this memory-inefficient approach requires 4KB of storage for t=1024 (assuming each index occupies 4 bytes). For microcontrollers like the ATmega328, which have limited flash memory and a threshold of 10,000 write/erase cycles, maintaining the index list in SRAM reduces flash write operations and preserves the data without corruption. Despite improving HORSE's [62] computationally expensive hash-based key generation (using Jakobsson hash calls [46]), this method still risks depleting public key chains (on verifier), potentially causing early termination.

8. Conclusion

As next-generation networks increasingly rely on heterogeneous IoTs with resource-limited devices, ensuring secure, scalable authentication is paramount. While traditional digital signatures provide necessary authentication tools, they face significant challenges due to the emerging threat of quantum computing and the inadequacies of current post-quantum cryptography (PQC) solutions for resource-constrained devices. To address these limitations, our Maximum Utilization Multiple HORS (MUM-HORS) signature scheme offers a lightweight, PQ-secure alternative tailored for IoT applications. With its ability to deliver fast signing, short signatures, and efficient key utilization, MUM-HORS provides significant improvements over existing solutions. The experimental results show its superior performance on embedded platforms, making it a promising candidate for secure, resource-efficient digital signatures in IoTs. Hence, MUM-HORS paves the way for more secure, scalable, and practical cryptographic solutions tailored to the evolving needs of applications like wearable medical devices or digital twins.

9. Acknowledgement

This research is partially supported by the Cisco Research Award (220159).

References Cited

[1] Abdul Ahad, Mohammad Tahir, Muhammad Aman Sheikh, Kazi Istiaque Ahmed, Amna Mughees, and Abdullah Numani. Technologies trend towards

- 5g network for smart health-care using iot: A review. *Sensors*, 20(14):4047, 2020.
- [2] Gorjan Alagic, Gorjan Alagic, Daniel Apon, David Cooper, Quynh Dang, Thinh Dang, John Kelsey, Jacob Lichtinger, Yi-Kai Liu, Carl Miller, et al. Status report on the third round of the nist post-quantum cryptography standardization process. 2022.
- [3] Moayad Aloqaily, Ouns Bouachir, Fakhri Karray, Ismaeel Al Ridhawi, and Abdulmotaleb El Saddik. Integrating digital twin and advanced intelligent technologies to realize the metaverse. *IEEE Consumer Electronics Magazine*, 12(6):47–55, 2022.
- [4] Apple. Apple Watch apple.com. https://www.apple.com/watch/, 2024. [Accessed 21-07-2024].
- [5] Giuseppe Ateniese, Giuseppe Bianchi, Angelo T Capossele, Chiara Petrioli, and Dora Spenza. Low-cost standard signatures for energy-harvesting wireless sensor networks. *ACM Transactions on Embedded Computing Systems* (*TECS*), 16(3):1–23, 2017.
- [6] Jean-Philippe Aumasson and Guillaume Endignoux. Clarifying the subsetresilience problem. *Cryptology ePrint Archive*, 2017.
- [7] Jean-Philippe Aumasson and Guillaume Endignoux. Improving stateless hash-based signatures. In *Cryptographers' Track at the RSA Conference*, pages 219–242. Springer, 2018.
- [8] Rouzbeh Behnia, Muslum Ozgur Ozmen, Attila A. Yavuz, and Mike Rosulek. Tachyon: Fast signatures from compact knapsack. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, pages 1855–1867, New York, NY, USA, 2018. ACM.
- [9] Rouzbeh Behnia and Attilla Altay Yavuz. Towards practical post-quantum signatures for resource-limited internet of things. In *Annual Computer Security Applications Conference*, ACSAC, page 119–130, New York, NY, USA, 2021. Association for Computing Machinery.
- [10] M. Bellare and P. Rogaway. Introduction to modern cryptography. In *UCSD CSE Course*, page 207. 1st edition, 2005. http://www.cs.ucsd.edu/~mihir/cse207/classnotes.html.

- [11] Daniel J. Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, and Bo-Yin Yang. High-speed high-security signatures. *Journal of Cryptographic Engineering*, 2(2):77–89, Sep 2012.
- [12] Daniel J Bernstein, Nadia Heninger, Paul Lou, and Luke Valenta. Post-quantum rsa. In *Post-Quantum Cryptography: 8th International Workshop, PQCrypto 2017, Utrecht, The Netherlands, June 26-28, 2017, Proceedings* 8, pages 311–329. Springer, 2017.
- [13] Daniel J. Bernstein, Daira Hopwood, Andreas Hülsing, Tanja Lange, Ruben Niederhagen, Louiza Papachristodoulou, Michael Schneider, Peter Schwabe, and Zooko Wilcox-O'Hearn. SPHINCS: Practical stateless hash-based signatures. In *Advances in Cryptology EUROCRYPT 2015: 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 368–397. Springer Berlin Heidelberg, April 2015.
- [14] Daniel J Bernstein, Daira Hopwood, Andreas Hülsing, Tanja Lange, Ruben Niederhagen, Louiza Papachristodoulou, Michael Schneider, Peter Schwabe, and Zooko Wilcox-O'Hearn. Sphincs: practical stateless hash-based signatures. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 368–397. Springer, 2015.
- [15] Daniel J. Bernstein, Andreas Hülsing, Stefan Kölbl, Ruben Niederhagen, Joost Rijneveld, and Peter Schwabe. The sphincs+ signature framework. Association for Computing Machinery, 2019.
- [16] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In Colin Boyd, editor, *Advances in Cryptology*—*ASIACRYPT 2001*, pages 514–532, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.
- [17] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT)*, 6(3):1–36, 2014.
- [18] Johannes Buchmann, Erik Dahmen, Sarah Ereth, Andreas Hülsing, and Markus Rückert. On the security of the winternitz one-time signature scheme. In *International conference on cryptology in Africa*, pages 363–378. Springer, 2011.

- [19] Johannes Buchmann, Erik Dahmen, and Andreas Hülsing. Xmss a practical forward secure signature scheme based on minimal security assumptions. In Bo-Yin Yang, editor, *Post-Quantum Cryptography*, pages 117–129. Springer Berlin Heidelberg, 2011.
- [20] Johannes Buchmann, Erik Dahmen, and Andreas Hülsing. Xmss a practical forward secure signature scheme based on minimal security assumptions. In *Proceedings of the 4th International Conference on Post-Quantum Cryptog-raphy*, PQCrypto'11, pages 117–129, Berlin, Heidelberg, 2011. Springer-Verlag.
- [21] Carmen Camara, Pedro Peris-Lopez, and Juan E Tapiador. Security and privacy issues in implantable medical devices: A comprehensive survey. *Journal of biomedical informatics*, 55:272–289, 2015.
- [22] Maria-Dolores Cano and Antonio Cañavate-Sanchez. Preserving data privacy in the internet of medical things using dual signature ecdsa. *Security and Communication Networks*, 2020(1):4960964, 2020.
- [23] Craig Costello and Patrick Longa. Schnorrq: Schnorr signatures on fourq. Technical report, MSR Tech Report, 2016. Available at: https://www.microsoft.com/en-us/research/wp-content/uploads/2016/07/SchnorrQ. pdf, 2016.
- [24] Nicolas T Courtois, Matthieu Finiasz, and Nicolas Sendrier. How to achieve a mceliece-based digital signature scheme. In *Advances in Cryptology—ASIACRYPT 2001: 7th International Conference on the Theory and Application of Cryptology and Information Security Gold Coast, Australia, December 9–13, 2001 Proceedings 7*, pages 157–174. Springer, 2001.
- [25] Post-Quantum Cryptography. Selected algorithms 2022. *URL: https://csrc.nist. gov/projects/post-quantum-cryptography/selected-algorithms-2022*, 2022.
- [26] Saleh Darzi, Kasra Ahmadi, Saeed Aghapour, Attila Altay Yavuz, and Mehran Mozaffari Kermani. Envisioning the future of cyber security in post-quantum era: A survey on pq standardization, applications, challenges and opportunities. *arXiv preprint arXiv:2310.12037*, 2023.
- [27] Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky. Lattice signatures and bimodal gaussians. In *Advances in Cryptology* -

- CRYPTO 2013 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I, pages 40–56, 2013.
- [28] Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky. Lattice signatures and bimodal gaussians. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology CRYPTO 2013: 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, pages 40–56, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.
- [29] Léo Ducas, Eike Kiltz, Tancrede Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-dilithium: A lattice-based digital signature scheme. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2018.
- [30] Léo Ducas, Vadim Lyubashevsky, and Thomas Prest. Efficient identity-based encryption over ntru lattices. In *Advances in Cryptology–ASIACRYPT* 2014: 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, ROC, December 7-11, 2014, Proceedings, Part II 20, pages 22–41. Springer, 2014.
- [31] Léo Ducas and Thomas Prest. Fast fourier orthogonalization. In *Proceedings* of the ACM on international symposium on symbolic and algebraic computation, pages 191–198, 2016.
- [32] Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-dilithium: A lattice-based digital signature scheme. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2018(1):238–268, Feb. 2018.
- [33] FiBit. Fitbit Official Site for Activity Trackers & More fitbit.com. https://www.fitbit.com/global/us/home, 2024. [Accessed 21-07-2024].
- [34] Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Prest, Thomas Ricosset, Gregor Seiler, William Whyte, Zhenfei Zhang, et al. Falcon: Fast-fourier lattice-based compact signatures over ntru. Submission to the NIST's post-quantum cryptography standardization process, 36(5):1–75, 2018.

- [35] Craig Gidney and Martin Ekerå. How to factor 2048 bit rsa integers in 8 hours using 20 million noisy qubits (2019). *arXiv preprint* arXiv:1905.09749, 1905.
- [36] M. Grissa, A. A. Yavuz, and B. Hamdaoui. Cuckoo filter-based location-privacy preservation in database-driven cognitive radio networks. In *Computer Networks and Information Security (WSCNIS)*, 2015 World Symposium on, pages 1–7, Sept 2015.
- [37] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, STOC '96, pages 212–219, New York, NY, USA, 1996. ACM.
- [38] Jorge Guajardo, Attila A. Yavuz, Benjamin Glas, Markus Ihle, Hamit Hacioglu, and Karsten Wehefrit. System and method for counter mode encrypted communication with reduced bandwidth. US 20140270163 A1, Filed: March 14, 2013, Issued: September 18, 2014.
- [39] Daniel Halperin, Thomas S Heydt-Benjamin, Kevin Fu, Tadayoshi Kohno, and William H Maisel. Security and privacy for implantable medical devices. *IEEE pervasive computing*, 7(1):30–39, 2008.
- [40] Thomas Häner, Samuel Jaques, Michael Naehrig, Martin Roetteler, and Mathias Soeken. Improved quantum circuits for elliptic curve discrete logarithms. In *Post-Quantum Cryptography: 11th International Conference, PQCrypto 2020, Paris, France, April 15–17, 2020, Proceedings 11*, pages 425–444. Springer, 2020.
- [41] Andreas Hülsing, Denis Butin, Stefan Gazdag, Joost Rijneveld, and Aziz Mohaisen. Xmss: extended merkle signature scheme, 2018.
- [42] Andreas Hülsing, Lea Rausch, and Johannes Buchmann. Optimal parameters for xmss mt. In Security Engineering and Intelligence Informatics: CD-ARES 2013 Workshops: MoCrySEn and SeCIHD, Regensburg, Germany, September 2-6, 2013. Proceedings 8, pages 194–208. Springer, 2013.
- [43] Andreas Hülsing, Lea Rausch, and Johannes Buchmann. Optimal parameters for xmss mt. In *International conference on availability, reliability, and security*, pages 194–208. Springer, 2013.

- [44] Andreas Hülsing, Joost Rijneveld, and Peter Schwabe. Armed sphincs. In *Proceedings, Part I, of the 19th IACR International Conference on Public-Key Cryptography PKC 2016 Volume 9614*, pages 446–470, Berlin, Heidelberg, 2016. Springer-Verlag.
- [45] Andreas Hülsing, Joost Rijneveld, and Peter Schwabe. Armed SPHINCS computing a 41 KB signature in 16 KB of RAM. In *Public-Key Cryptog-raphy PKC 2016 19th IACR International Conference on Practice and Theory in Public-Key Cryptography*, pages 446–470, March 2016.
- [46] Markus Jakobsson. Fractal hash sequence representation and traversal. In *Proceedings IEEE International Symposium on Information Theory*, page 437. IEEE, 2002.
- [47] Don Johnson, Alfred Menezes, and Scott Vanstone. The elliptic curve digital signature algorithm (ecdsa). *International Journal of Information Security*, 1(1):36–63, Aug 2001.
- [48] Simon Josefsson and Ilari Liusvaara. Edwards-curve digital signature algorithm (eddsa). Technical report, 2017.
- [49] Matthias J Kannwischer, Joost Rijneveld, Peter Schwabe, and Ko Stoffelen. pqm4: Testing and benchmarking nist pqc on arm cortex-m4. 2019.
- [50] Syed Hussain Ali Kazmi, Rosilah Hassan, Faizan Qamar, Kashif Nisar, and Ag Asri Ag Ibrahim. Security concepts in emerging 6g communication: Threats, countermeasures, authentication techniques and research directions. *Symmetry*, 15(6):1147, 2023.
- [51] Leslie Lamport. Constructing digital signatures from a one-way function. Technical report, Technical Report CSL-98, SRI International Palo Alto, 1979.
- [52] Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *Designs, Codes and Cryptography*, 75(3):565–599, 2015.
- [53] Hakjun Lee. Slars: Secure lightweight authentication for roaming service in smart city. *IEICE Transactions on Communications*, 2024.

- [54] J. Lee, S. Kim, Y. Cho, Y. Chung, and Y. Park. HORSIC: An efficient one-time signature scheme for wireless sensor networks. *Information Processing Letters*, 112(20):783 787, 2012.
- [55] Jaeheung Lee and Yongsu Park. Horsic+: An efficient post-quantum few-time signature scheme. *Applied Sciences*, 11(16):7350, 2021.
- [56] Vadim Lyubashevsky. Fiat-shamir with aborts: Applications to lattice and factoring-based signatures. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 598–616. Springer, 2009.
- [57] David McGrew, Michael Curcio, and Scott Fluhrer. Hash-Based Signatures. Internet-Draft draft-mcgrew-hash-sigs-15, Internet Engineering Task Force, January 2019. Work in Progress.
- [58] David McGrew, Panos Kampanakis, Scott Fluhrer, Stefan-Lukas Gazdag, Denis Butin, and Johannes Buchmann. State management for hash-based signatures. In *Security Standardisation Research: Third International Conference, SSR 2016, Gaithersburg, MD, USA, December 5–6, 2016, Proceedings 3*, pages 244–260. Springer, 2016.
- [59] Frank McKeen, Ilya Alexandrovich, Ittai Anati, Dror Caspi, Simon Johnson, Rebekah Leslie-Hurd, and Carlos Rozas. Intel® software guard extensions (intel® sgx) support for dynamic memory management inside an enclave. In *Proceedings of the Hardware and Architectural Support for Security and Privacy 2016*, HASP 2016, New York, NY, USA, 2016. Association for Computing Machinery.
- [60] Abolfazl Mehbodniya, Julian L Webber, Rahul Neware, Farrukh Arslan, Raja Varma Pamba, and Mohammad Shabaz. Modified lamport merkle digital signature blockchain framework for authentication of internet of things healthcare data. *Expert Systems*, 39(10):e12978, 2022.
- [61] Ralph Charles Merkle. *Secrecy, authentication, and public key systems.* Stanford university, 1979.
- [62] William D Neumann. Horse: an extension of an r-time signature scheme with fast signing and verification. In *International Conference on Information Technology: Coding and Computing*, 2004.

- [63] Dinh C Nguyen, Ming Ding, Pubudu N Pathirana, Aruna Seneviratne, Jun Li, Dusit Niyato, Octavia Dobre, and H Vincent Poor. 6g internet of things: A comprehensive survey. *IEEE Internet of Things Journal*, 9(1):359–383, 2021.
- [64] Saif E Nouma and Attila A Yavuz. Post-quantum forward-secure signatures with hardware-support for internet of things. In *ICC 2023-IEEE International Conference on Communications*, pages 4540–4545. IEEE, 2023.
- [65] Saif E. Nouma and Attila A. Yavuz. Practical cryptographic forensic tools for lightweight internet of things and cold storage systems. IoTDI '23, page 340–353, New York, NY, USA, 2023. Association for Computing Machinery.
- [66] Saif E Nouma and Attila A Yavuz. Trustworthy and efficient digital twins in post-quantum era with hybrid hardware-assisted signatures. *ACM Transactions on Multimedia Computing, Communications and Applications*, 2023.
- [67] Aleksandr Ometov, Pavel Masek, Lukas Malina, Roman Florea, Jiri Hosek, Sergey Andreev, Jan Hajny, Jussi Niutanen, and Yevgeni Koucheryavy. Feasibility characterization of cryptographic primitives for constrained (wearable) iot devices. In 2016 IEEE International Conference on Pervasive Computing and Communication Workshops. IEEE, 2016.
- [68] Muslum Ozgur Ozmen, Attila A Yavuz, and Rouzbeh Behnia. Energy-aware digital signatures for embedded medical devices. In 2019 IEEE Conference on Communications and Network Security (CNS), pages 55–63. IEEE, 2019.
- [69] Renuka Sahebrao Pawar and Dhananjay Ramrao Kalbande. Optimization of quality of service using eceba protocol in wireless body area network. *International Journal of Information Technology*, 15(2):595–610, 2023.
- [70] A. Perrig. The BiBa: One-time signature and broadcast authentication protocol. In *Proceedings of the ACM Conference on Computer and Communications Security*, pages 28–37, November 2001.
- [71] Krzysztof Piotrowski, Peter Langendoerfer, and Steffen Peter. How public key cryptography influences wireless sensor node lifetime. In *Proceedings of the fourth ACM on Security of ad hoc and sensor networks*, 2006.

- [72] Manas Pradhan and Josef Noll. Security, privacy, and dependability evaluation in verification and validation life cycles for military iot systems. *IEEE Communications Magazine*, 58(8):14–20, 2020.
- [73] John Proos and Christof Zalka. Shor's discrete logarithm quantum algorithm for elliptic curves. *arXiv preprint quant-ph/0301141*, 2003.
- [74] Martin Raab and Angelika Steger. "balls into bins"—a simple and tight analysis. In *International Workshop on Randomization and Approximation Techniques in Computer Science*, pages 159–170. Springer, 1998.
- [75] Leonid Reyzin and Natan Reyzin. Better than biba: Short one-time signatures with fast signing and verifying. Springer, 2002.
- [76] R.L. Rivest, A. Shamir, and L.A. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.
- [77] Sebastian Rohde, Thomas Eisenbarth, Erik Dahmen, Johannes Buchmann, and Christof Paar. Fast hash-based signatures on constrained devices. In *Smart Card Research and Advanced Applications: 8th IFIP WG 8.8/11.2 International Conference, CARDIS 2008.* Springer, 2008.
- [78] Mikail Mohammed Salim, Laurence Tianruo Yang, and Jong Hyuk Park. Lightweight authentication scheme for iot based e-healthcare service communication. *IEEE Journal of Biomedical and Health Informatics*, 2023.
- [79] Rajkumar SC, Karthick KR, et al. Secure session key pairing and a lightweight key authentication scheme for liable drone services. *Cyber Security and Applications*, 1:100012, 2023.
- [80] C. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.
- [81] Adi Shamir and Yael Tauman. Improved online/offline signature schemes. In Advances in Cryptology—CRYPTO 2001: 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19–23, 2001 Proceedings 21, pages 355–367. Springer, 2001.
- [82] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Review*, 1999.

- [83] D. Stinson. *Cryptography: Theory and Practice, Second Edition*. CRC/C&H, 2002.
- [84] Mehdi Tibouchi and Alexandre Wallet. One bit is all it takes: a devastating timing attack on bliss's non-constant time sign flips. *Journal of Mathematical Cryptology*, 15(1):131–142, 2020.
- [85] Jo Van Bulck, David Oswald, Eduard Marin, Abdulla Aldoseri, Flavio D Garcia, and Frank Piessens. A tale of two worlds: Assessing the vulnerability of enclave shielding runtimes. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 1741–1758, 2019.
- [86] Qiyan Wang, Himanshu Khurana, Ying Huang, and Klara Nahrstedt. Time valid one-time signature for time-critical multicast data authentication. In *IEEE INFOCOM 2009*, pages 1233–1241. IEEE, 2009.
- [87] Xu Yang, Xun Yi, Surya Nepal, Ibrahim Khalil, Xinyi Huang, and Jian Shen. Efficient and anonymous authentication for healthcare service with cloud based wbans. *IEEE Transactions on Services Computing*, 15(5):2728–2741, 2021.
- [88] A. A. Yavuz, A. Mudgerikar, A. Singla, I. Papapanagiotou, and E. Bertino. Real-time digital signatures for time-critical networks. *IEEE Transactions on Information Forensics and Security*, 2017.
- [89] A. A. Yavuz and M. O. Ozmen. Ultra lightweight multiple-time digital signature for the internet of things devices. *IEEE Transactions on Services Computing*, pages 1–1, 2019.
- [90] Attila A. Yavuz. ETA: Efficient and tiny and authentication for heterogeneous wireless systems. In *Proceedings of the sixth ACM conference on Security and privacy in wireless and mobile networks*. ACM.
- [91] Attila A. Yavuz. An efficient real-time broadcast authentication scheme for command and control messages. *Information Forensics and Security, IEEE Transactions on*, 9(10):1733–1742, Oct 2014.
- [92] Attila A Yavuz, Saleh Darzi, and Saif E Nouma. Lightweight and scalable post-quantum authentication for medical internet of things.

- [93] Attila A. Yavuz, Peng Ning, and Michael K. Reiter. BAF and FI-BAF: Efficient and publicly verifiable cryptographic schemes for secure logging in resource-constrained systems. *ACM Transaction on Information System Security*, 15(2), 2012.
- [94] G.M. Zaverucha and D.R. Stinson. Short one-time signatures. Cryptology ePrint Archive, Report 2010/446, 2010. https://eprint.iacr.org/2010/446.

Appendix A. Bitmap Functionalities Using Linked List

This section details the implementation of BM functionalities using a linked list instead of a circular queue and examines the resulting performance changes. The same design applies to managing public keys on the verifier. This design change affects only the key generation phase of MUM-HORS while signing and verification proceed unchanged. Consequently, we present the key generation for MUM-HORS and the full details of the new implementation of the BM.

Appendix A.1. MUM-HORS Key Generation and Bitmap Operations

Based on Algorithm 3, key generation changes affect Steps 2 and 3. In Step 2, BM is initialized as a linked list with rt nodes, where head and tail are pointers to the first and last nodes, and each row has a next parameter pointing to the subsequent row. In Step 3, public keys are generated similarly as a r-node linked list, with each node containing t public keys derived from the master key msk. Note that the verifier does not need to load the entire public key list into memory to avoid overhead. Instead, public keys are stored on disk, and only rt nodes are loaded into memory as needed. Both signer and verifier share the semantics of the BM parameters for synchronization as before.

Algorithm 3 Key Generation of MUM-HORS (Linked List Version)

 $(sk, PK, \text{BM}, I_{\text{MUM-HORS}}) \leftarrow \text{MUM-HORS.Kg}(1^{\kappa})$:

- 1: Set $I_{\text{MUM-HORS}} \leftarrow (I_{\text{HORS}}, r, rt), msk \stackrel{\$}{\leftarrow} \{0, 1\}^{\kappa}$, and $\{pad_i\}_{i=1}^3$
- 2: Create bitmap $BM = \{row_i\}_{i=1}^{rt}$ and $set\ row_i.num = i,\ row_i.active bits = t,\ row_i.next = row_{i+1},\ all\ row_i.bits[.]\ to\ 1,\ and\ row_{rt}.next = \bot.$ Set $BM.head = row_1,\ BM.tail = row_{rt},\ BM.window = t,\ BM.nextrow = rt+1,\ BM.active rows = rt,\ and\ BM.active bits = rt \times t.$
- 3: Create public keys $PK \leftarrow \{pk_i\}_{i=1}^r$ and set $pk_i.num = i$, $pk_i.activepks = t$, $pk_i.next = pk_{i+1}$, $\{pk_i.keys[j] \leftarrow f(PRF(msk||i||j))\}_{1 \leq i \leq r, 1 \leq j \leq t}$, and $pk_r.next = \bot$. Set $PK.head = pk_1$, $PK.tail = pk_{rt}$, PK.window = t, PK.nextrow = rt + 1, PK.activerows = rt, and $PK.activepks = rt \times t$.
- 4: Send the private key $sk \leftarrow msk$, $\{pad_i\}_{i=1}^3$, and BM to the signer, store the public keys PK and $\{pad_i\}_{i=1}^3$ on the verifier, and output the $I_{\texttt{MUM-HORS}}$

The BM operations using a linked list are detailed in Algorithm 4. The function BITMAP_EXTEND_MATRIX(.) extends the BM if there are insufficient set bits (less than window bits), calling BITMAP_CLEANUP(.) to remove rows with zero active bits or, if necessary, the row with the fewest active bits. Fresh rows are then added to the end of the list, which is pointed to by the tail parameter. The BITMAP_GET_ROW_COLUMN(.) function returns the row and column indices of

the $(index + 1)^{th}$ set bit for the given index by traversing the list instead of a circular queue. The BITMAP_UNSET_INDICES(.) function unsets the provided indices where for each index index, the $(index + 1)^{th}$ set bit of the window is set to zero, with the main difference being the traversal method.

Algorithm 4 Bitmap Manipulation Operations (Linked List Version)

```
b \leftarrow \texttt{BITMAP\_EXTEND\_MATRIX}(\texttt{BM}):
1: if BM.activebits < BM.window then
      if {\tt BM}.nextrow > r then return {\tt False}
2:
3:
      if BITMAP_CLEANUP(BM) == 0 then
4:
          Set row to the row with the minimum active bits parameter
5:
          Update BM.activerows = 1 and BM.activebits = row.activebits
6:
          Remove row from BM
7:
       fill capacity = min(rt - BM.activerows, r - BM.nextrow)
       Add fill capacity rows to BM by creating new row and setting row.num = BM.nextrow,
   row.activebits = t, all row.bits[.] to 1, row.next = \bot, BM.tail.next = row and
   BM.nextrow += 1 for each.
9: return True
   n \leftarrow \texttt{BITMAP\_CLEANUP}(\texttt{BM}): Set cleaned = 0 and row = \texttt{BM}.head
1: while row is not \perp do
2:
       if row.activebits == 0 then
          Remove row from BM and set cleaned += 1 and row = row.next
3:
4: return cleaned
   (row, col) \leftarrow \texttt{BITMAP\_GET\_ROW\_COLUMN}(\texttt{BM}, index) : \textbf{Set } row = \texttt{BM}.head
1: while row is not \perp do
2:
       if index < row.active bits then break
       Update index = row.activebits and row = row.next
4: Set colidx as the index of (index + 1)^{th} set bit in row.bits[.] and return (row.num, colidx)
   BITMAP_UNSET_INDICES(BM, indices):
1: for index in indices do
2:
      row = BM.head
       while row is not \perp do
3:
          if index < row.active bits then break
4:
5:
          index = row.activebits and row = row.next
      Unset the (index + 1)^{th} bit in row.bits[.] to 0
6:
       Update BM.activebits = 1 and row.activebits = 1
```

Appendix A.2. Performance Evaluation and Comparison

Using a linked list instead of a circular queue adds an 8-byte next field per node, increasing memory usage by 88 bytes for (rt = 11) rows on a 64-bit sys-

tem. Although linked lists enable faster iteration compared to shifting rows in a circular queue—particularly for middle rows—the additional memory overhead is a tradeoff for reduced computational complexity. On a commodity hardware BITMAP_CLEANUP(.) takes $0.027\mu s$, BITMAP_UNSET_INDICES(.) $0.18\mu s$ per index, and BITMAP_GET_ROW_COLUMN(.) $0.23\mu s$. On Raspberry Pi 4, the performance is: BITMAP_CLEANUP(.) takes $0.51\mu s$, BITMAP_UNSET_INDICES(.) $0.77\mu s$ per index, and BITMAP_GET_ROW_COLUMN(.) $1.47\mu s$. The array-based implementation incurs additional computation due to row shifting and slightly higher costs for row iteration due to being a circular queue.

Appendix B. MUM-HORS Signer and Verifier Index Synchronization

Given the challenge of state synchronization in MTSs built on OTSs [58], MUM-HORS is vulnerable to desynchronization if messages are corrupted in transit. To address this, we propose the Second Chance Algorithm (SCA), allowing the verifier to recover from out-of-sync states by retaining mismatched public keys. This self-correcting approach enables the verifier to restore synchronization using future valid message-signature pairs without requiring communication with the signer. The new MUM-HORS verifier is given in Algorithm 5.

The first two steps and steps 11-12 are similar to the MUM-HORS's verifier presented in Algorithm 1. SCA algorithm allows each rejected public key to remain in the list for a second chance, as the rejection cause (message or signature corruption) may be unclear. Normal public keys are denoted as pk_{ij} and those given a second chance as pk_{ij}^* . Using the property of HORS key indices (distance between each key), we apply the following rules: (i) For a verified pair (s_i, s_j) with i < j, if there are j - i public keys between them, mark all pk_{ij}^* in between as pk_{ij} . (ii) For a verified pair (s_i, s_{i+1}) , mark all pk_{ij}^* in between as \perp (deleted). If a signature cannot be validated, we not only mark the pk_{ij} as pk_{ij}^* but also delete the actual value of the public key for the received private key and mark it as \perp if the signature is not corrupted (Steps 3-10).

It is important to note that, according to the security proof in Section 6, the advantage in forging a signature does not arise from corrupting the signature along with the message, as its security relies on the pre-image resistance of the one-way function used. Therefore, we can assume that any signature corruption results from transmission errors, which can be corrected using error correction codes. As a result, the SCA algorithm can be optimized under the assumption of message corruption. However, the strict rules of the SCA algorithm may hinder immediate

Algorithm 5 MUM-HORS Verifier with SCA algorithm

```
b \leftarrow \texttt{MUM-HORS.Ver}(PK, m, \sigma): Set hash \leftarrow Trunc(H(m), k \log t) and b = 1
1: if hash or any pad_{1,2,3} yield distinct i_j as steps 1-2 of MUM-HORS. Sig(.), then goto 3
 2: Split hash' \leftarrow Trunc(H(hash||Ctr), k \log t) as step 1 of MUM-HORS.Sig(.) into inte-
    ger indices i_j. if \exists p,q \in [1,k] s.t. i_p = i_q and p \neq q then Execute steps 3-4 of
    MUM-HORS.Sig(.) and return b=0.
3: for j = 1 to k do
        Find the (i_i)^{th} public key as in <code>BITMAP_GET_ROW_COLUMN(.)</code> and count all the pk_{ij}^*
    until that point as doubt
5:
        if s_i is verified then mark the corresponding public key as \perp
6:
7:
            Mark the s_i's public key as pk_{ij}^* and try the next doubt non-deleted public keys.
8:
            if any verified then check with last verified signature (s_{j'}) given j > j'. if there are
    j-j' public keys then mark all the pk_{ij}^* between s_{j'} and s_j as pk_{ij}, else if j'=j+1, then,
    mark all the the pk_{ij}^* as the \perp.
9:
            else return b=0
10: return b
                                              - Verifier is Idle
11: Invalidate all the pk_j corresponding to the derived i_j as in BITMAP_UNSET_INDICES(.)
12: Extend the view of the public keys similar to BITMAP_EXTEND_MATRIX(.)
```

state recovery during many consecutive corruptions, as the number of corruptions could exceed j - j' between two indices i_j and $i_{j'}$.

The SCA algorithm provides a probabilistic solution to the synchronization issue. However, the most reliable approach is to reset both the signer and verifier to a fresh row number—flushing the bitmap and verifier's memory—once the number of corruptions exceeds a specified threshold. This can be managed through an additional communication mechanism and software checks. It is worthy to note this fact again that the synchronization issue is inherent to stateful HB signatures and is not only limited to our scheme [58].