Fast and Post-Quantum Authentication for Real-time Next Generation Networks with Bloom Filter

Kiarash Sedghighadikolaei
Computer Science and Engineering
University of South Florida
Tampa, FL, USA
kiarashs@usf.edu

Attila A Yavuz
Computer Science and Engineering
University of South Florida
Tampa, FL, USA
attilaayavuz@usf.edu

***Abstract*—Large-scale next-generation networked systems like smart grids and vehicular networks facilitate extensive automation and autonomy through real-time communication of sensitive messages. Digital signatures are vital for such applications since they offer scalable broadcast authentication with non-repudiation. Yet, even conventional secure signatures (e.g., ECDSA, RSA) introduce significant cryptographic delays that can disrupt the safety of such delay-aware systems. With the rise of quantum computers breaking conventional intractability problems, these traditional cryptosystems must be replaced with post-quantum (PQ) secure ones. However, PQ-secure signatures are significantly costlier than their conventional counterparts, vastly exacerbating delay hurdles for real-time applications.**

We propose a new signature called *Time Valid Probabilistic Data Structure HORS (`TVPD-HORS`)* that achieves significantly lower end-to-end delay with a tunable PQ-security for real-time applications. We harness special probabilistic data structures as an efficient one-way function at the heart of our novelty, thereby vastly fastening `HORS` as a primitive for NIST PQ cryptography standards. `TVPD-HORS` permits tunable and fast processing for varying input sizes via One-hash Bloom Filter, excelling in time valid cases, wherein authentication with shorter security parameters is used for short-lived yet safety-critical messages. We show that `TVPD-HORS` verification is 2.7× and 5× faster than `HORS` in high-security and time valid settings, respectively. `TVPD-HORS` key generation is also faster, with a similar signing speed to `HORS`. Moreover, `TVPD-HORS` can increase the speed of `HORS` variants over a magnitude of time. These features make `TVPD-HORS` an ideal primitive to raise high-speed time valid versions of PQ-safe standards like XMSS and SPHINCS+, paving the way for real-time authentication of next-generation networks.

***Index Terms*—Internet of Things; post-quantum security; digital signature; next-generation networks; Bloom filter**

I. Introduction

The Internet of Things (IoT) and mobile cyber-physical systems rely on Next-Generation (NextG) networks to enable large-scale autonomy through real-time communication among network entities. Smart-grids [23] and the Internet of Vehicles are some examples in which real-time communication is essential to maintain the reliability of the application. For instance, in smart grids, the timely verification of command and control messages and fast yet accurate analysis of measurements (e.g., from smart meters) are vital to avoid cascade failures and damages [19], [32]. Similarly, vehicular network standards [1] emphasize the importance of delay awareness for safe operation for autonomous driving [3], [11].

The trustworthiness of such delay-aware applications requires that security-sensitive real-time communication is authenticated and integrity-protected. Digital signatures permit scalable broadcast authentication with non-repudiation and public verifiability and are foundational tools for NextG networked systems. For example, some vehicular communication standards require broadcasting several digital signatures per second (e.g., ECDSA [21]) to enable secure vehicular communication [1]. NISTIR 7628 [32] recommends digital signatures in smart grids for authentication. However, the overhead of digital signatures may negatively impact the reliability of delay-aware applications [16]. For instance, ECDSA introduces a significant end-to-end delay, impacting functionalities like timely break [38], similar concerns applicable for potential cascade failures in smart grids with stringent delay requirements (e.g., a few msecs [32]).

The emergence of quantum computers, capable of breaking conventional secure signatures based on the discrete logarithm and integer factorization problems using Shor's algorithm [35], necessitates the adoption of post-quantum (PQ) safe alternatives. Despite the selection of three schemes—CRYSTALS-Dilithium [14], FALCON [15], and SPHINCS+ [7]—for standardization by NIST in response to this threat [2], [13], current PQ-safe signatures are significantly more costly than conventional secure ones, exacerbating the risks of cryptographic delays. For instance, SPHINCS+ is several magnitudes and a magnitude slower than ECDSA in signing and verifying, respectively. Similarly, Dilithium and Falcon are also much more computationally costlier than their conventional-secure NIST standards (and even more compared to their faster variants such as Ed25519 [6], FourQ [9]). Moreover, as shown in [22], Falcon-512 requires 117KB, Dilithium requires 113KB, and SPHINCS+ requires 9KB (excluding its very large signature being around 31 KB) for both code and stack memory on ARM Cortex-M4, which represents a significant overhead for resource-constrained devices. Hence, delay-aware systems should prioritize post-quantum security solutions that ensure security for short durations while maintaining computational and communication efficiency. We further discuss related work and research challenges in section VI.

A. Our Contribution

We created a new lightweight (one-time) PQ secure signature called *Time Valid Probabilistic Data Structure HORS* (TVPD−HORS) to fill the need for a fast and security-adaptable digital signature for delay-aware applications. At the core of our innovation lies harnessing PDS with special features to attain rapid and tunable OWF, thereby enhancing the performance of HORS. Specifically, we synergies One-hash Bloom Filter (OHBF) [27], which requires only a single size-compatible hash call with a small-constant number of modular arithmetic, with precise parameter tuning. We enhanced our scheme by incorporating weak message resilience. We outline some of TVPD−HORS's desirable properties below:

- *Fast Signature Verification:* TVPD−HORS offers 2.7× and up to 5× faster signature verification for high (up to 128-bit) and time valid security (between 32-64 bit) parameters, respectively, against HORS with standard-compliant SHA256/512, and with similar performance advantages if size-speed optimized Blake family used in time valid cases. These performance advantages persist against HORS variants with standard cryptographic hashes, for example, over magnitude speed differences versus HORSIC [24] and HORSIC+ [25].
- *Fast Key Generation with Equal Signing Performance:* TVPD−HORS key generation is 6× and up to 1.3× faster than that of HORS with SHA256/512 for time valid and high-security settings, respectively, also being up to 4× faster for the time valid case against HORS with Blake. The signing speed and signature size of TVPD−HORS are the same as with HORS. Therefore, TVPD−HORS offers the lowest end-to-end delay among its counterparts thanks to its faster verification and key generation with a similar signing speed.
- *A Fast and Tunable PQ-Secure Building Block:* One-time TVPD−HORS outperforms HORS in almost all settings but especially in time valid cases. Therefore, it is a suitable candidate to serve as a building block for PQ-secure standards like XMSS and SPHINCS+ that rely on HORS variants. In particular, TVPD−HORS is ideal for constructing time valid versions of these PQ-secure standards to support real-time applications via one-time to multiple-time transformations.
- *Full-fledge Implementation:* We fully implemented TVPD−HORS scheme on a commodity hardware available at:

https://github.com/kiarashsedghigh/tvpdhors

II. Preliminaries and Models

Notations: $||$ and $|x|$ denote concatenation and the bit length of x, respectively. $x \xleftarrow{\$} \mathcal{S}$ 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 integer values from 1 to n. $f : \{0,1\}^* \rightarrow \{0,1\}^\kappa$ is an OWF. $H : \{0,1\}^* \rightarrow \{0,1\}^L$ and $h : \{0,1\}^* \rightarrow \{0,1\}^{L'}$ denote cryptographic and non-cryptographic hash functions, respectively.

Definition 1 A one-time hash-based digital signature SGN consists of three algorithms:

- $(sk, PK, I_{SGN}) \leftarrow \texttt{SGN.Kg}(1^\kappa)$: Given the security parameter κ, it outputs the private key sk, the public key PK, and the system-wide parameters I_{SGN}.
- $\sigma \leftarrow \texttt{SGN.Sig}(sk, m)$: Given the private key sk and message m, it returns the signature σ.
- $b \leftarrow \texttt{SGN.Ver}(PK, m, \sigma)$: 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) [33] is a hash-based digital signature consists of three algorithms:

- $(sk, PK, I_{\texttt{HORS}}) \leftarrow \texttt{HORS.Kg}(1^\kappa)$: Given the security parameter κ, it selects $I_{\texttt{HORS}} \leftarrow (t, k, l)$, 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 $PK \leftarrow \{v_i\}_{i=1}^{t}$.
- $\sigma \leftarrow \texttt{HORS.Sig}(sk, m)$: 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 $\sigma \leftarrow \{s_{i_j}\}_{j=1}^{k}$.
- $b \leftarrow \texttt{HORS.Ver}(PK, m, \sigma)$: Given PK, m, and σ, it computes $\{i_j\}_{j=1}^{k}$ as in HORS.Sig(.). If $v_{i_j} = f(\sigma_j), \forall j = 1, \ldots, k$, it returns $b = 1$, otherwise $b = 0$.

Definition 3 A Probabilistic Data Structure (PDS) [8] for a set $\mathcal{U}$ comprises at least three algorithms:

- $(bv, \overrightarrow{h}, I_{\texttt{PDS}}) \leftarrow \texttt{PDS.Init}(1^\kappa)$: Given the security parameter κ, it selects $I_{\texttt{PDS}} \leftarrow (n, k)$, creates a zeroed n-bit vector $bv[.]$ and samples k hash functions $\overrightarrow{h} \leftarrow \{h_i\}_{i=1}^{k}$. It then outputs $(bv[.], \overrightarrow{h}, I_{\texttt{PDS}})$.
- $\texttt{PDS.Insert}(bv, \overrightarrow{h}, u)$: Given bv, $\overrightarrow{h}$, and $u \in \mathcal{U}$, it computes the indices $\{i_j\}_{j=1}^{k} \leftarrow f(\overrightarrow{h}, u)$ and sets $bv[i_j] = 1$ $\forall j = 1, \ldots, k$. $f()$ is design dependent.
- $b \leftarrow \texttt{PDS.Check}(bv, \overrightarrow{h}, u)$: Given bv, $\overrightarrow{h}$, and u, it computes the indices $\{i_j\}_{j=1}^{k} \leftarrow f(\overrightarrow{h}, u)$ and checks if $bv[i_j] = 1$ $\forall j = 1, \ldots, k$. If so, it returns $b = 1$, meaning u was probably inserted before, and $b = 0$ otherwise.

Definition 4 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 $\leq T$:

$$\text{InSec}_{\mathcal{H}}^{RSR}(T) = \max_{\mathcal{A}}\{\Pr[(M_1, M_2, ..., M_{r+1}) \leftarrow \mathcal{A}(i,t,k) \text{ s.t. } H_{i,t,k,L}(M_{r+1}) \subseteq \bigcup_{j=1}^{r} H_{i,t,k,L}(M_j)]\} < \text{negl}(t,k)$$

$$\text{InSec}_{\mathcal{H}}^{SPR}(T) = \max_{\mathcal{A}}\{\Pr[x \leftarrow \{0,1\}^*; x' \leftarrow \mathcal{A}(x) \text{ s.t. } x \neq x' \text{ and } H_{i,t,k,L}(x) = H_{i,t,k,L}(x')\} < \text{negl}(L)$$

Definition 5 Let $\mathcal{PDS} = \{\texttt{PDS}_{i,n,k,L'}\}$ be a function family indexed by i, where $\texttt{PDS}_{i,n,k,L'}$ has an n-bit vector and k L'-bit hash functions. $\mathcal{PDS}$ is collision-resistant (CR) and one-way (OW) if, for every PPT $\mathcal{A}$ running in time $\leq T$:

$$\text{InSec}^{CR}_{\mathcal{PDS}}(T) = \max_{\mathcal{A}}\{\Pr[u \leftarrow \mathcal{A}(\texttt{PDS}_{i,n,k,L'}) \text{ s.t. } u \text{ was not inserted before and } \texttt{PDS.Check}(u) = 1]\} < \text{negl}(n,k,L')$$

The CR property denotes the false positive probability.

$$\text{InSec}^{OW}_{\mathcal{PDS}}(T) = \max_{\mathcal{A}}\{\Pr[u \leftarrow \mathcal{A}(\texttt{PDS}_{i,n,k,L'}) \text{ s.t. } u \text{ was inserted before and } \texttt{PDS.Check}(u) = 1]\} < \text{negl}(n,k,L')$$

System Model: We assume a broadcast environment [38], in which the signer sends security-sensitive messages to be authenticated by verifiers. TVPD-HORS is designed for delay-aware applications in which timely verification is vital, like smart grids, wherein several low-end peripheral devices (e.g., smart meters) periodically upload their telemetry to a cloud-supported state monitoring system for immediate authentication [23]. TVPD-HORS is lightweight and suitable for low-end devices (e.g., 8-bit microcontrollers). However, we assume the verifiers can store several public keys (e.g., a cloud server).

We consider *time valid* delay-aware applications, wherein the security of some transmitted messages remains critical only for a specific time interval[1](see Section VI). Our scheme also preserves its performance advantage in high-security parameters but is especially performant on moderate-level security, making it ideal for time valid applications.

Threat and Security Model: Our threat model assumes an adversary $\mathcal{A}$ that can monitor all message-signature pairs and aims to intercept, modify, and forge them. $\mathcal{A}$ is quantum computing capable and aims at forgery in a designated time interval to succeed. The digital signature security model capturing our threat model follows the Existential Unforgeability under Chosen Message Attacks (*EU-CMA*).

Definition 6 The one-time *EU-CMA* (*OEU-CMA*) experiment for one-time signature SGN is defined as follows:

- $(sk, PK, I_{\text{SGN}}) \leftarrow \texttt{SGN.Kg}(1^\kappa)$
- $(m^*, \sigma^*) \leftarrow \mathcal{A}^{\texttt{SGN.Sig}_{sk}(.)}(PK, I_{\text{SGN}})$
- $\mathcal{A}$ wins the experiment with a maximum of one query allowed if $1 \leftarrow \texttt{SGN.Ver}(PK, m^*, \sigma^*)$ and m^* was not queried to the signing oracle $\texttt{SGN.Sig}_{sk}(.)$.

$$Succ^{OEU\text{-}CMA}_{\text{SGN}}(\mathcal{A}) = Pr[Expt^{OEU\text{-}CMA}(\mathcal{A}) = 1]$$
$$InSec^{OEU\text{-}CMA}_{\text{SGN}}(T) = \max_{\mathcal{A}}\{Succ^{OEU\text{-}CMA}_{\text{SGN}}(\mathcal{A})\} < negl(T)$$

III. Proposed Scheme

We propose TVPD-HORS that synergizes special PDS with efficient hash functions to enable overall high performance with significantly faster operations in time valid settings. Recall that using standard BF with HORS (e.g., [34]) yields highly inefficient results due to excessive hash calls to maintain false positives at par with security parameters. We overcome this challenge by adapting the One-hash Bloom Filter (OHBF) [27] that utilizes only one efficient hash operation combined with fast modulo operations suitably selected for the target security levels. Finally, we introduce weak key mitigation [4] into TVPD-HORS often omitted in its counterparts. We describe TVPD-HORS in Algorithm 1 and elaborate it as follows:

TVPD-HORS.Kg(.) sets system-wide parameters $I_{\text{TVPD-HORS}}$, comprising HORS parameters (Definition 2), OHBF parameters (n, p), and time epoch parameter T_Δ for potential time valid settings (Step 1). Next, a zeroed n-bit vector $bv[.]$ with p partitions of each of size n_i, satisfying $\Sigma n_i \geq n$ and $gcd(n_i, n_j) = 1$ for all $i, j \in [1, p]$, is created [27] (done as $\texttt{PDS.Init}(n, k = 1)$) (Step 2). The HORS private key is generated (Step 3), and each element s_i is inserted, along with its index i, into the $bv[.]$ by setting the bit of every partition $j \in [1, p]$ at index $(h(s_i||i) \mod n_j)$ (Step 4). Time synchronization parameters are set if a valid setting is selected (Step 5). This step is skipped for high-security levels (e.g., 72-bit to 128-bit security).

TVPD-HORS.Sig(.) resembles the HORS signing. In time-valid settings, signing occurs only in a given time slice T_Δ, while at high-security levels, no such restriction is imposed (Step 1). To eliminate weak message vulnerability, m is concatenated with counter Ctr to ensure its hash contains k distinct $\log t$-sized parts (Steps 2-6). Since only $k \log t$ bits of the hash are used (security reduction from L-bit to $k \log t$-bits),

Algorithm 1 Time Valid Probabilistic Data Structure HORS

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

1: Set $I_{\text{TVPD-HORS}} \leftarrow (I_{\text{HORS}}, n, p, T_\Delta)$
2: Create zeroed $bv[.]$ having p partitions $\{P_i\}_{i=1}^{p}$ each of size n_i
3: Generate t random l-bit strings $\{s_i\}_{i=1}^{t}$: $s_i \xleftarrow{\$} \{0,1\}^l, \forall i \in [1, t]$
4: Insert s_i into the $bv[.]$ by setting the bit $(h(s_i||i) \mod n_j)$ of j^{th} partition to 1, $\forall i \in [1, t]$ and $\forall j \in [1, p]$
5: **if** time valid setting **then** set T_s and T_v to T_0 // T_Δ depends on κ and application needs (e.g., ranging from a minute to days).
6: **return** the private key $sk \leftarrow \{s_i\}_{i=1}^{t}$, the public key $PK \leftarrow bv[.]$, and the system-wide parameters $I_{\text{TVPD-HORS}}$

$\sigma \leftarrow \texttt{TVPD-HORS.Sig}(sk, m)$: $Ctr \leftarrow 0$ continue as follows:

1: **if** κ is high security level **or** $T_s \in [T_0, T_0 + T_\Delta]$
2: $hash \leftarrow H(m||Ctr)$
3: $hash' \leftarrow Trunc(hash, k \log t)$
4: Split $hash'$ into k substrings $\{hash'_j\}_{j=1}^{k}$ s.t. $|hash'_j| = \log t$
5: Interpret each $hash'_j$ as an integer i_j, $\forall j \in [1, k]$
6: **if** there are $p, q \in [1, k]$ s.t. $i_p = i_q$ and $p \neq q$ **then** $Ctr \leftarrow Ctr + 1$ and **goto** step 2
7: **return** $\sigma \leftarrow (\{s_{i_j}\}_{j=1}^{k}, Ctr)$

$b \leftarrow \texttt{TVPD-HORS.Ver}(PK, m, \sigma)$:

1: **if** κ is not high security level **and** $T_v \notin [T_0, T_0 + T_\Delta]$ **then return** $b = 0$
2: $hash \leftarrow H(m||Ctr)$
3: $hash' \leftarrow Trunc(hash, k \log t)$
4: Split $hash'$ into k substrings $\{hash'_j\}_{j=1}^{k}$ s.t. $|hash'_j| = \log t$
5: Interpret each $hash'_j$ as an integer i_j, $\forall j \in [1, k]$
6: **if** $\exists\ p, q \in [1, k]$ s.t. $i_p = i_q$ and $p \neq q$ **then return** $b = 0$
7: **if** all the bit indices $(h(s'_j||i_j) \mod n_i)$ of the i^{th} partition in the $bv[.]$ are set, $\forall j \in [1, k]$, $\forall i \in [1, p]$ **then return** $b = 1$ **else return** $b = 0$

[1]We indicate time valid forgery attacks targeting temporal (real-time) messages but not the long-term attacks aiming at components like master certificates in public key infrastructures.

the hash output is truncated to $k \log t$ (Step 3).

TVPD−HORS.Ver(.) first checks if the verification occurs within its designated time interval (Step 1) in time-valid settings (skipped for high-security levels). Next, it checks weak message conditions with the received Ctr by ensuring $hash'$ contains k distinct parts (Steps 2-6). Finally, it verifies if the signature elements (s'_j) are valid by checking the existence of $(s'_j||i_j)$ in the $bv[.]$ (done as PDS.Check(.)) (Step 7). If all the signature elements exist, then the signature is valid.

A. One-time Key Management for Longevity, Storage, and Scalability

It is necessary to consider the implications of time-bounding the generation and use of TVPD−HORS public keys while binding them with long-term secure certificates. We consider two practical directions when TVPD−HORS is extended from OTS to a scalable N-time signature: (i) N TVPD−HORS public keys are derived from a seed [39], masked with a random pad, and stored on the verifier. During each signing round, the signer derives the signature and pad, while the verifier unmasks the public key using the pad, and verification proceeds as in TVPD−HORS. This method provides long-term protection via certificates, sustainable signature services, and long-term security through masking. (ii) Utilizing secure enclaves via Intel SGX [28] as explored in [30] can eliminate the need for signers to provide commitments and certificates. This approach may extend OTSs for multiple signatures and faster key generation. In this method, the verifier generates the public key using the master key in a secure enclave and then loads it into main memory for verification, as in TVPD−HORS.

IV. Performance Analysis And Comparison

We first present a comprehensive performance comparison of TVPD−HORS with HORS for time valid and full-security settings and then compare TVPD−HORS with other HORS variants to showcase its potential.

A. Evaluation Metrics and Experimental Setup

We evaluate private/public key and signature sizes, and then key generation (done offline), signature generation, and verification times for all compared schemes. We implemented compared schemes on a desktop with an Intel i9-11900K@3.5GHz processor and 64GB of RAM. We used (i) LibTomCrypt[2] to implement SHA2(256/512), (ii) Blake2[3], and (iii) CityHash[4] and xxHash3[5] as non-cryptographic hash functions.

B. Parameter Selection

Selecting parameters for a specific security level involves considering: (1) the hash function for message hashing, (2) $k \log t$ bits of the hash output, (3) HORS signature security $k(\log t - \log k)$, (4) OHBF hash collision security, and (5) OHBF false positive probability. Grover's algorithm [18] can reverse a black-box function with input size N in $O(\sqrt{N})$ steps and $O(\log_2 N)$ qubits. We use this model to evaluate the security of our hash function for message signing and the OHBF hash function. Thus, a hash function with L-bit output provides $\frac{L}{2}$ bits of security against quantum adversaries.

Based on [27], the false positive probability (fpp) of OHBF is calculated as follows, where k denotes the number of partitions, n the number of elements inserted, and m_i the size of the i^{th} partition:

$$fpp = \left(1 - \sqrt[k]{\prod_{i=1}^{k} e^{-\frac{n}{m_i}}}\right)^k \tag{1}$$

To adapt this to our signature, we replaced n with t, renamed k to p, and m_i to n_i to avoid parameter conflicts. We used Algorithm 1 from [27] to determine the partition size n_i, with a Python3 implementation available in our repository [6]. Using the algorithm, partition sizes are derived by specifying the total OHBF size in bits, the number of partitions, and the number of elements. Once partitions are set, the false positive probability can be calculated as given above.

Regarding parameter selection, for instance, to achieve 32-bit security, by selecting $t = 64$ and $k = 16$ as in TABLE I, the SHA2-256 hash function provides $\frac{256}{2}$-bit security, but only $k \log t = 96$ bits are covered by HORS, reducing the hash security to 48 bits. The HORS security is $k(\log t - \log k) = 32$ bits. Using xxHash3-64 as the OHBF hash function provides 32-bit security. Therefore, the minimum security guarantee is 32 bits, which also requires OHBF's false positive probability to be 32-bit. Using Algorithm 1 from [27], we experimentally, with different sizes and numbers of partitions, obtained $p = 8$ with a 995-byte OHBF and $p = 6$ with a 1915-byte OHBF. We selected the first setting to keep public keys smaller, although the second option may be preferred if storage is less of a concern. The parameters yield the following partition sizes in bits, which are further used to insert the keys:

Partitions = [971, 977, 983, 991, 997, 1009, 1013, 1019]

Note that TVPD−HORS includes a built-in partition calculator implemented in C; the Python3 script is provided for demonstration purposes only. The complete parameter list can be found at our code repository [7].

C. Efficiency Evaluation and Comparison

Our comparison spans various security levels, ranging from 32-bit to 64-bit for time valid applications and 72-bit to 128-bit for non-time-valid applications (medium/high-security). Security levels are adjusted based on the underlying primitives and their parameters (e.g., SHA2-256/512, CityHash-256, xxHash3-64, p in the OHBF, etc.)

In Tables Ia-Ib, we compare TVPD−HORS with HORS when various hash functions are used as $f()$ or $h()$, for varying security levels in time valid (κ=32, 48, and 64) and high-security (κ=72, 96, and 128) settings. We first instantiated

[2]https://github.com/libtom/libtomcrypt
[3]https://github.com/BLAKE2/
[4]https://github.com/google/cityhash
[5]https://xxhash.com/

[6]https://github.com/kiarashsedghigh/tvpdhors/blob/main/misc/ohbf.py
[7]https://github.com/kiarashsedghigh/tvpdhors/blob/main/misc/Parameters.png

TABLE I: Performance comparison of TVPD−HORS and HORS

(a) TVPD−HORS vs HORS ($f()$: SHA2 family)

Scheme	(t, k, l, p)	PK (KB)	Kg (μs)	Ver (μs)	κ
HORS	(64, 16, 32, 8)	2	12.08	3.11	32
TVPD−HORS		0.971	1.99	0.66	
HORS	(64, 32, 32, 8)	2	13.14	6.32	32
TVPD−HORS		0.971	1.96	1.28	
HORS	(128, 16, 48, 17)	4	24.99	3.15	48
TVPD−HORS		1.87	10.45	0.709	
HORS	(256, 16, 64, 28)	8	48.08	3.13	64
TVPD−HORS		3.93	30.84	0.647	
HORS	(128, 32, 64, 28)	4	24.03	6.19	64
TVPD−HORS		1.95	16.23	1.3	
HORS	(512, 16, 72, 36)	16	98.37	3.17	72
TVPD−HORS		7.9	87.96	1.12	
HORS	(256, 32, 96, 38)	8	46.89	6.01	96
TVPD−HORS		5.44	45.81	2.19	
HORS	(512, 32, 128, 28)	16	94.35	6.01	128
TVPD−HORS		40.73	72.48	2.29	
HORS	(256, 64, 128, 30)	8	47.26	11.83	128
TVPD−HORS		17.58	37.66	4.37	

In all settings, $f()$ of HORS was SHA2-256 and $h()$ of TVPD−HORS was xxHash3-64 for 32-bit, xxHash3-128 for (48,64)-bit and CityHash-256 for (72,128)-bit levels.

(b) TVPD−HORS vs HORS ($f()$: Blake2 family)

Scheme	(t, k, l, p)	PK (KB)	Kg (μs)	Ver (μs)	κ
HORS	(64, 16, 32, 8)	1	8.51	1.85	32
TVPD−HORS		0.971	1.99	0.66	
HORS	(64, 32, 32, 8)	1	7.92	3.69	32
TVPD−HORS		0.971	1.96	1.28	
HORS	(128, 16, 48, 17)	2	15.17	1.81	48
TVPD−HORS		1.87	10.45	0.709	
HORS	(256, 16, 64, 28)	4	29.55	1.85	64
TVPD−HORS		3.93	30.84	0.647	
HORS	(128, 32, 64, 28)	2	15.28	3.65	64
TVPD−HORS		1.95	16.23	1.3	
HORS	(512, 16, 72, 30)	10	58.02	1.91	72
TVPD−HORS		6.28	76.5	1.14	
HORS	(256, 32, 96, 32)	8	29.02	3.68	96
TVPD−HORS		6.34	40.09	2.22	
HORS	(512, 32, 128, 28)	16	71.52	4.65	128
TVPD−HORS		40.73	72.48	2.29	
HORS	(256, 64, 128, 30)	8	36.22	9.12	128
TVPD−HORS		17.58	37.66	4.37	

$f()$ of HORS was Blake2s-128 for (32-64)-bit, Blake2s-160 for 72-bit, and Blake2b-256 for (96-128)-bit. $h()$ of TVPD−HORS was xxHash3-64 for 32-bit, xxHash3-128 for 48 and 64-bit and CityHash-256 for (72,128)-bit levels.

We used SHA-256 for $H()$ in all cases except 128-bit security level with SHA-512. Message size is 256 Bytes in all cases. The private key size is $t \cdot l$ for both schemes, but it can also be extracted from a constant-size seed. The PK column for TVPD−HORS shows the size of the OHBF (parameter n). The signature size is $k \cdot l$ plus $\log |Ctr|$ variable (negligible) for both schemes. The signing time is also the same since all signing functionalities are identical.

HORS with SHA2-256 to show its performance with NIST compliance and then with the Blake family to offer a speed-optimized time valid comparison against TVPD−HORS. We used the xxHash3 and the CityHash families as $h()$ for TVPD−HORS. We outline our findings as follows:

(i) The verification of TVPD−HORS is 3-5× and 2.7× faster than HORS with standard-compliant SHA-256 (TABLE Ia) in time valid and high-security parameters, respectively. TVPD−HORS is also 2.8× and 2× faster than HORS with speed-size optimized Blake variants in time valid and high-security parameters, respectively (TABLE Ib). (ii) TVPD−HORS key generation shows 1.5-6× improvement for time valid settings and up to 1.3× for high-security levels with standard-compliant SHA-256 (TABLE Ia). Moreover, TVPD−HORS is 4.2× faster for 32-bit and 48-bit security levels with comparable performance for other levels with Blake (TABLE Ib). (iii) The PK size of TVPD−HORS is smaller than that of HORS for all security levels except 128-bit, for which we opted for a larger key for faster key generation. Thanks to OHBF, TVPD−HORS permits a more flexible PK size trade-off, and we can choose smaller public key sizes with slower key generation (it is offline). (iv) The signing performance is the same for both schemes.

Tables IIa-IIb offer a comprehensive performance comparison of TVPD−HORS with other prominent HORS variants analytically (asymptotic) and experimentally (estimated with $f()$ as SHA-256 for $\kappa = 32, 64, 128$), respectively: (i) The signature verification of HORSE, HORS++, and TV-HORS resembles that of HORS. However, HORST requires authentication of each private key element s_i using a Merkle tree, making its verification more expensive than others. In HORSIC and HORSIC+, parameters z and w impact the signature verification, respectively. With the provided configurations, TVPD−HORS exhibits a 3-32× improvement in time valid settings and a 1.5-24× improvement in high-security levels. (ii) While HORSE and HORS++ have similar key generation as HORS, HORST needs a Merkle tree on top of private key elements s_i to derive the public key, doubling the cost. In HORSIC and HORSIC+, the parameter w significantly impacts key generation. Opting for small w with minimal impact on HORSIC and HORSIC+, TVPD−HORS still delivers a 6-13× improvement for time valid settings and 1.2-3× improvement at the 128-bit security level. (iii) HORST, HORSE, HORS++, and TV-HORS exhibit similar signature generation as HORS, while HORSIC and HORSIC+ have costlier signing. All variants consider weak message mitigation with an extra cost explained in TABLE IIa. (iv) Compared to Shafieinejad et al. [34] who used standard bloom filter as OWF, TVPD−HORS shows 20-28× faster key generation and 14-50× faster signature verification in time-valid settings and 4× and 9×, respectively, in 128-bit setting.

In summary, some notable takeaways are: (i) The verification speed of TVPD−HORS surpasses that of HORS with high-security, and with a significantly growing performance advantage in time valid settings. (ii) The key generation of TVPD−HORS is faster than HORS in all levels with standard-compliant hash and remains comparable or slightly lesser in size-adjusted speed-optimized hashes for HORS. Note that key generation is mostly done offline, and we can provide size-speed trade-offs to fasten key generation. (iii) The signing performance of TVPD−HORS is the same as that of HORS. (iv) The performance superiority of TVPD−HORS over HORS also remains valid if not grown in various HORS variants. In comparison to NIST's PQ-secure standards for a 128-bit security level, signature verification with TVPD−HORS is 103× faster than SPHINCS+, 15× faster than Dilithium, and 8× faster

TABLE II: Performance comparison of TVPD−HORS and HORS variants

(a) Analytical (asymptotic) comparison results

Scheme	sk Size	PK Size	Signature Size	Key Generation	Signature Generation	Signature Verification
HORS (t,k,l) [33]	$t{\cdot}l$	$t{\cdot}\lvert f()\rvert$	$k{\cdot}l{+}\log\lvert Ctr\rvert$	$t{\cdot}f()$	$k{\cdot}O(1){+}\mu{\cdot}H()$	$H(){+}k{\cdot}f()$
HORST (t,k,l) [7]	$t{\cdot}l{+}(t-1){\cdot}\lvert f()\rvert$	$\lvert f()\rvert$	$(k+\log t){\cdot}\lvert f()\rvert{+}\log\lvert Ctr\rvert$	$(2t-1){\cdot}f()$	$k{\cdot}O(1){+}\mu{\cdot}H()$	$H(){+}k(\log t{+}1){\cdot}f()$
HORSE (t,k,l,d) [29]	$t{\cdot}l{+}t{\cdot}(d-1){\cdot}\lvert f()\rvert$	$t{\cdot}\lvert f()\rvert$	$k{\cdot}l{+}\log\lvert Ctr\rvert$	$t{\cdot}d{\cdot}f()$	$k{\cdot}O(1){+}\mu{\cdot}H()$	$H(){+}k{\cdot}f()$
HORS++ (t,k,l) [31]	$t{\cdot}l$	$t{\cdot}\lvert f()\rvert$	$k{\cdot}l{+}\log\lvert Ctr\rvert$	$t{\cdot}f()$	$k{\cdot}O(1){+}\mu{\cdot}H()$	$H(){+}k{\cdot}f()$
HORSIC (t,k,l,z,w) [24]	$t{\cdot}(l{+}w{\cdot}\lvert f()\rvert)$	$t{\cdot}\lvert f()\rvert$	$k{\cdot}\lvert f()\rvert{+}\log\lvert Ctr\rvert$	$w{\cdot}t{\cdot}f()$	$k{\cdot}w{\cdot}f(){+}\mu{\cdot}H(){+}G(){+}C_{k,z}()$	$H(){+}(z+2){\cdot}f(){+}G(){+}C_{k,z}()$
HORSIC+ (n,t,k,l,z,w) [25]	$w{\cdot}l{+}t{\cdot}(l{+}w{\cdot}\lvert f()\rvert)$	$(1+w+t){\cdot}\lvert f()\rvert$	$k{\cdot}\lvert f()\rvert{+}\log\lvert Ctr\rvert$	$w{\cdot}t{\cdot}f()$	$k{\cdot}w{\cdot}f(){+}\mu{\cdot}H(){+}G(){+}C_{k,z}()$	$H(){+}k{\cdot}w{\cdot}f(){+}G(){+}C_{k,z}()$
Shafieinejad et al. (t,k,l,n,m) [34]	$t{\cdot}l$	m	$k{\cdot}l{+}\log\lvert Ctr\rvert$	$t{\cdot}n{\cdot}h()$	$k{\cdot}O(1){+}\mu{\cdot}H()$	$H(){+}k{\cdot}n{\cdot}h()$
TV-HORS (t,k,l,T_Δ,T_ϕ) [36]	$\lceil\frac{T_\phi}{T_\Delta}\rceil{\cdot}(l{+}t{\cdot}\lvert f()\rvert)$	$(l{+}t{\cdot}\lvert f()\rvert){+}O(1)$	$k{\cdot}\lvert f()\rvert{+}l{+}\log\lvert Ctr\rvert{+}O(1)$	$\lceil\frac{T_\phi}{T_\Delta}\rceil{\cdot}t{\cdot}f()$	$k{\cdot}O(1){+}\mu{\cdot}H()$	$H(){+}k{\cdot}f(){+}O(1)$
TVPD−HORS (t,k,l,n,p)	$t{\cdot}l$	$\sum_{i=1}^{p}n_i$	$k{\cdot}l+\log\lvert Ctr\rvert$	$t{\cdot}(h(){+}p{\cdot}O(1))$	$k{\cdot}O(1){+}\mu{\cdot}H()$	$H(){+}k{\cdot}(h(){+}p{\cdot}O(1))$

The memory complexity of the private key is the memory expansion caused by deriving the private keys from a constant-sized master key. The average value of the message-dependent μ is $\frac{t^k}{t(t-1)\ldots(t-k)}$. Hash function $G()$ and the bijective function $C_{k,z}()$ are specific to HORSIC and HORSIC+. Although HORS variants are different in their design when used as a one-time signature, their parameters may be ineffective, such as d set to 1 in HORSE, and hence, they perform like HORS.

(b) Experimental performance comparison results ($f()$: SHA2-256, $h()$: xxHash3-(64,128)/CityHash-256)

Scheme	sk Size (KB)			PK Size (KB)			Sig Size (KB)			Key Gen (μs)			Sig Gen (μs)			Sig Ver (μs)		
	$\kappa=32$	$\kappa=64$	$\kappa=128$	$\kappa=32$	$\kappa=64$	$\kappa=128$	$\kappa=32$	$\kappa=64$	$\kappa=128$	$\kappa=32$	$\kappa=64$	$\kappa=128$	$\kappa=32$	$\kappa=64$	$\kappa=128$	$\kappa=32$	$\kappa=64$	$\kappa=128$
HORS (t,k,l) [33]	0.25	1	4	2	4	8	0.06	0.25	1	12.08	24.03	47.26	890.9	893.18	901.02	3.41	6.49	12.13
HORST (t,k,l) [7]	2.21	4.96	11.96	0.31			0.68	1.21	2.25	25.82	51.53	101.83	891.12	893.98	901.42	23.17	52.03	115.08
HORSE (t,k,l,d) [29]	0.25	1	4	2	4	8	0.06	0.25	1	12.05	23.87	48.89	890.97	893.34	900.88	3.28	6.35	11.85
HORS++ (t,k,l) [31]	0.25	1	4	2	4	8	0.06	0.25	1	11.92	24.44	47.78	891.21	893.76	901.76	3.45	6.50	12.32
HORSIC (t,k,l,z,w) [24]	4.25	9	20	2	4	8	0.34	0.56	1	26.03	51.73	102.03	895.97	899.19	904.25	3.14	4.74	7.27
HORSIC+ (n,t,k,l,z,w) [25]	4.257	9.015	20.03	2	4	16	0.17	0.28	1	26.15	52.12	103.66	898.54	899.97	904.49	5.01	7.42	12.51
Shafieinejad et al. (t,k,l,n,m) [34]	0.25	1	4	0.968	1.97	17.57	0.06	0.25	1	56.32	345.6	168.95	891.59	893.61	901.42	14.12	86.5	42.23
TV-HORS (t,k,l,T_Δ,T_ϕ) [36]	2.003	4.007	8.015	2.015	4.019	20.03	0.515	1.015	2.015	12.23	23.89	47.15	891.33	893.24	901.18	3.79	6.81	12.41
TVPD−HORS (t,k,l,n,p)	**0.25**	**1**	**4**	**0.97**	**1.95**	17.58	**0.06**	**0.25**	**1**	**1.96**	**16.23**	**37.66**	**891.33**	**893.42**	**901.54**	**0.96**	**1.6**	**4.67**

Message size is 256 Bytes. For **HORS**, **HORST**, **HORSE**, **HORS++**, and **TV-HORS**, the parameters (t,k,l) have been set to (64, 16, 32) for $\kappa=32$, (128, 32, 64) for $\kappa=64$, and (256, 64, 128) for $\kappa=128$. The parameter d of HORSE and the ratio $\lceil\frac{T_\phi}{T_\Delta}\rceil$ of TV-HORS have been set to 1 as we are using them as one-time signature. For **HORSIC** the parameters (t,k,l,z,w) have been set to (64, 11, 32, 12, 2) for $\kappa=32$, (128, 19, 64, 20, 2) for $\kappa=64$, and (256, 32, 128, 33, 2) for $\kappa=128$. For **HORSIC+** with $n=256$ the parameters (t,k,l,z,w) have been set to (64, 11, 32, 12, 2) for $\kappa=32$, (128, 18, 64, 19, 2) for $\kappa=64$, and (256, 32, 128, 33, 2) for $\kappa=128$. For **Shafieinejad et al.** [34], the parameters (t,k,l,n,m) where n and m denote the number of hashes and size of the Bloom Filter, respectively, have been set to (64, 16, 32, 8, 0.96KB) for $\kappa=32$, (128, 32, 64, 27, 1.97KB) for $\kappa=64$ and (256, 64, 128, 30, 17.57KB) for $\kappa=128$. Moreover, xxHash3-64 was used for $\kappa=32$, xxHash3-128 for $\kappa=64$ and CityHash256 for $\kappa=128$. For **TVPD−HORS**, the parameters of (t,k,l,n,p) have been set to (64, 16, 32, 8) for $\kappa=32$, (128, 32, 64, 28) for $\kappa=64$, and (256, 64, 128, 30) for $\kappa=128$.

than Falcon. Additionally, as shown in [22], for code and stack, Falcon-512 requires 117KB, Dilithium 113KB, and SPHINCS+ 9KB of storage on ARM Cortex-M4, making them impractical for resource-constrained devices. (v) End-to-end speed advantage, especially due to faster verification, makes TVPD−HORS suitable for real-time applications, and its performance with other HORS variants demonstrate its potential for advancing multiple-time signatures such as XMSS and SPHINCS+ in time valid settings.

V. Security Analysis

Theorem 1 *TVPD−HORS is OEU-CMA secure if $H()$ is r-subset-resilient and second-preimage resistant, and OHBF is collision resistant and one-way:*

$$\mathit{InSec}^{\mathit{OEU\text{-}CMA}}_{\mathit{TVPD\text{-}HORS}}(T) = \mathit{InSec}^{\mathit{RSR}}_{H}(T) + \mathit{InSec}^{\mathit{SPR}}_{H}(T) + \mathit{InSec}^{\mathit{CR}}_{\mathit{OHBF}}(T) + \mathit{InSec}^{\mathit{OW}}_{\mathit{OHBF}}(T) < \mathit{negl}(t,k,n,p,L,L')$$

Proof: Given valid message-signature pair (m,σ), there are the below cases leading to a forgery:

- *$\mathcal{A}$ breaks r-subset-resilient of H*: $\mathcal{A}$ finds m^* such that $H(m^*)$ has the same k distinct elements as $H(m)$ but $H(m^*)\neq H(m)$. The success probability of $\mathcal{A}$ is $(\frac{k}{t})^k$, which denotes that after k elements determined by $H(m)$, the k elements of $H(m^*)$ are a subset of them and is negligible for appropriate values of k and t. Depending on m, $H(m)$ may lack k distinct elements, which are called weak messages. They increase the forgery probability as $\mathcal{A}$ may require fewer s_i elements from the sk, which might be found in the current signature σ. To ensure $H(m)$ possesses k distinct elements, we employ the incremental variable Ctr in concatenation with the message, generating the desired hash as shown in Steps 2-6 of TVPD−HORS.Sig(.).
- *$\mathcal{A}$ breaks the second-preimage resistance of H*: $\mathcal{A}$ can find m^* such that $H(m)=H(m^*)$ and output valid (m^*,σ). As H is anf L-bit cryptographic hash function, the probability of finding such collision is $\frac{1}{2^{\frac{L}{2}}}$. In addition, the selection of parameters k and t impacts the security of H. As in HORS, the condition $k\log t = L$ must hold. If $k\log t < L$, then the success probability of $\mathcal{A}$ increases from $\frac{1}{2^{\frac{L}{2}}}$ to $\frac{1}{2^{\frac{k\log t}{2}}}$. Therefore, the success probability of $\mathcal{A}$ is $max(\frac{1}{2^{\frac{L}{2}}},\frac{1}{2^{\frac{k\log t}{2}}})$. In TVPD−HORS, to ensure $k\log t = L$, we truncate the message's hash to the size $k\log t$ using the $Trunc(.)$ function.
- *$\mathcal{A}$ finds collision on OHBF*: $\mathcal{A}$ generates $\sigma^* \leftarrow \{r_i\}_{i=1}^{k}$, where r_i is a random value, on m^*, such that when the verifier checks the existence of signature elements as $\{r_j||i_j\}_{j=1}^{k}$ (where i_j is derived as Step 5 in TVPD−HORS.Ver(.)), it outputs 1, indicating that the signature is valid. That is, $\mathcal{A}$ tries to find r_i such that when concatenated with their index from the truncated value of $H(m^*||Ctr)$, step 7 of TVPD−HORS.Ver(·) returns 1. Given OHBF as the

underlying PDS with p partitions $\{P_i\}_{i=1}^{p}$ each of size n_i, the collision probability (false positive probability) of inserting N items is $(1-\sqrt[p]{\Pi_{i=1}^{p} e^{-\frac{N}{n_i}}})^{-p}$ [27]. Moreover, as $h()$ is an L'-bit hash function, the collision probability based on the Birthday paradox is $\frac{1}{2^{\frac{L'}{2}}}$. Hence, the advantage of $\mathcal{A}$ is $max(\frac{1}{2^{\frac{L'}{2}}}, (1-\sqrt[p]{\Pi_{i=1}^{p} e^{-\frac{t}{n_i}}})^{p})$ given $N=t$.

- *$\mathcal{A}$ inverts OHBF*: Given $bv[.]$, $\mathcal{A}$ recovers the secret key elements $\{s_i\}_{i=1}^{t}$ inserted into the $bv[.]$ as $\{s_i||i\}_{i=1}^{t}$ during TVPD-HORS.Kg(.) (with sk, $\mathcal{A}$ can forge signature on any message). Let $Hashes = \{\bigcap_{i=1}^{p}\{x \text{ s.t. } bv[x \mod n_i] = 1, \forall i \in [1,p]\}\}$ as the set of all possible $h()$'s output x who caused the bits $bv[.]$ to be set. Given that OHBF is using L'-bit $h()$ and based on the Birthday paradox, the probability of finding t distinct s_i such that $\{s_i||i\}$ hashes to a value in $Hashes$ is $\frac{1}{2^{\frac{L'}{2}}}$. Therefore, the success probability of recovering the whole secret key will be $(\frac{1}{2^{\frac{L'}{2}}})^t$. However, $\mathcal{A}$ does not need the entire secret key but only k elements of it such that $\{s_j||i_j\}_{j=1}^{k}$ (where i_j is derived as Step 5 in TVPD-HORS.Ver(.)) hashes to a value in $Hashes$. Therefore, the probability is $(\frac{1}{2^{\frac{L'}{2}}})^k$. This case closely resembles the previous case where OHBF is not collision-resistant as $\mathcal{A}$ can distinguish the new s_i from the actual secret key element with advantage of $\frac{1}{2^{\frac{L'}{2}}}$. Overall, we conclude:

$$\text{InSec}^{\text{OEU-CMA}}_{\text{TVPD-HORS}}(T) = max(\frac{1}{2^{\frac{L}{2}}}, \frac{1}{2^{\frac{k\log t}{2}}}) + (\frac{k}{t})^k + (\frac{1}{2^{\frac{L'}{2}}})^k + max(\frac{1}{2^{\frac{L'}{2}}}, (1-\sqrt[p]{\Pi_{i=1}^{p} e^{-\frac{t}{n_i}}})^{p}) < \text{negl}(t,k,L,L',n,p)$$

VI. Related Work and Research Challenges to be Addressed

Standard signatures based on Elliptic Curve Cryptography (ECC) [6], [21] are mentioned in smart-grid [32], 5G, and vehicular standards [1] with an expressed need for faster alternatives. Various high-speed signatures proposed (e.g., [38]), but they mostly rely on conventional intractability assumptions (e.g., (EC) Discrete Log Problem, factorization), which can be broken by quantum computers [35].

NIST Post-Quantum Cryptography (NIST-PQC) standard [10], which includes lattice-based (e.g., Dilithium [14]) and hash-based (HB) (SPHINCS+ [7]) alternatives, offer quantum-safe signatures. Despite their merits, these general-purpose signatures are not suitable for delay-aware applications since they are costlier than their conventional-secure counterparts. In particular, HB standards (XMSS [20], SPHINCS+ [7]) offer high PQ-security without relying on any number-theoretical assumption making them preferable for security-critical applications. Yet, they introduce high signing and verification overhead. Hence, creating delay-aware HB signatures is a valuable research direction, and we aim to do so by enhancing their underlying building blocks and enabling tunable security and performance trade-offs.

Quest for Efficient One-way Functions (OWF) for Faster HB Signatures: One of the most efficient HB one-time signature schemes is Hash-to-Obtain Random Subset (HORS) [33], known for its computational efficiency, also serving as the building block for XMSS and SPHINCS+. Several HORS variants are proposed (e.g., [25], [26], [29], [31], [36]) relying on various chaining techniques, time valid security, or offering extended functionalities on top [41].

The verification and key generation overhead of HORS is dominated by one-way functions (OWFs) typically implemented with SHA-256/512 [12]. While standard cryptographic hashes are efficient for general-purpose applications, they incur substantial computational overhead when invoked in mass, as in XMSS or SPHINCS+. Therefore, an often omitted but crucial means to enhance HORS like signatures is to identify, improve, and integrate efficient OWFs into their design.

Time Valid Security and Need for Tunable OWFs: In a time valid application, security-sensitive yet short-lived messages need authentication and integrity only for relatively short time durations [36]. For instance, real-time command and telemetry for rapid decision-making in a smart-grid system permit the adversary only a brief amount of time (e.g., a few seconds) to forge their corresponding signature. Otherwise, the adversary misses her opportunity to influence the system as the target message has already been processed. In such cases, timely verification of the short-lived message is the priority, which raises a time valid design relying on private/public key pairs with shorter security parameters and lifespan.

HB-signatures are ideal for time valid schemes [36], since unlike lattice-based signatures [5], [14] with various mutually dependent parameters, one can only adjust the length of the hash function with a smooth security response. Yet, the lack of efficient OWFs with variable (small) input sizes is an obstacle to building time valid HB schemes. For instance, consider a time valid HORS that aims security levels $32 \leq \kappa \leq 72$ depending on the application requirements, where κ (bit) denotes the security parameter. For $\kappa = 40$, only 40-bit private key elements are inputted to OWF. Regardless of the length of the short input, SHA256 always performs the same amount of computation; in this example, it processes 40 bits with the cost of a full 256 bits. One may try to mitigate the waste via non-standard lightweight cryptographic hash functions like Blake-128; however, they still lack tunable and fast OWF capabilities as they inherently process fixed block sizes.

In this work, we harness and optimize probabilistic data structures (PDS) to attain better OWF efficiency for HORS. Note that Bloom-filters (BFs) are often used for privacy-enhancing technologies (e.g., [17]), but to a lesser extent for authentication purposes (e.g., [37], [40]). The closest (and to our knowledge only) alternative to our work is in [34], which suggests using a basic BF in HB signatures. However, we identify that a straightforward use of BF in HORS does not improve but worsens the performance. This is due to BF's false positive (FP) rate impacting that of HORS. Even FP rates approaching only low-to-moderate security levels (e.g., $\kappa = 64$) require an excessive number of BF (non-cryptographic) hash calls, incurring an overhead significantly more than just implementing HORS with SHA256 as OWF.

There is a significant need for novel signature building

blocks that can offer fast and tunable performance to meet the stringent delay requirements of real-time NextG networked systems in the post-quantum era.

VII. Conclusion

Next-generation networked systems, like smart grids and vehicular networks, rely on real-time communication to enable automation and autonomy. However, conventional-secure cryptosystems are vulnerable to emerging quantum computers, while existing PQ signatures are significantly costlier than their traditional counterparts, exacerbating delay and reliability hurdles. In response to these challenges, we propose a novel signature scheme called *Time Valid Probabilistic Data Structure HORS* (TVPD−HORS), designed to achieve significantly lower end-to-end delays while offering tunable PQ security. We innovate on HORS by introducing special PDS with minimal overhead, speed-optimized hashes with fine-grained parameterization, and weak-message countermeasures. TVPD−HORS achieves 3-5× and 2.7× faster verification in time-valid and high-security settings, respectively, compared to HORS with SHA256. The speed advantages of TVPD−HORS remain valid against HORS using lightweight hashes, with identical signing speed and comparable key sizes to ensure low end-to-end delay in all cases. TVPD−HORS also offers over-magnitude speed gains when used with HORS variants, showcasing its versatility to support various HB signatures. Our results indicate that TVPD−HORS has a significant potential to fasten HB signature standards and serve as an ideal building block to secure NextG real-time networks.

VIII. Acknowledgment

This research is partially supported by the NSF CNS-2350213 grant and Cisco Research Award (220159).

References Cited

[1] Ieee approved draft standard for wireless access in vehicular environments–security services for applications and management messages. *IEEE Std 1609.2-2022 (Revision of IEEE Std 1609.2-2016)*, 2023.

[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] John Anderson, Qiqing Huang, Long Cheng, and Hongxin Hu. A zero trust architecture for connected and autonomous vehicles. *IEEE Internet Computing*, 2023.

[4] Jean-Philippe Aumasson and Guillaume Endignoux. Clarifying the subset-resilience problem. *Cryptology ePrint Archive*, 2017.

[5] 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.

[6] 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.

[7] Daniel J. Bernstein, Andreas Hülsing, Stefan Kölbl, Ruben Niederhagen, Joost Rijneveld, and Peter Schwabe. The sphincs+ signature framework. Association for Computing Machinery, 2019.

[8] Burton H Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.

[9] 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.

[10] Post-Quantum Cryptography. Selected algorithms 2022. *URL: https://csrc. nist. gov/projects/post-quantum-cryptography/selected-algorithms-2022*, 2022.

[11] Josh Dafoe, Harsh Singh, Niusen Chen, and Bo Chen. Enabling real-time restoration of compromised ecu firmware in connected and autonomous vehicles. In *International Conference on Security and Privacy in Cyber-Physical Systems and Smart Vehicles*, pages 15–33. Springer, 2023.

[12] Quynh Dang. Secure hash standard, 2015-08-04 2015.

[13] 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.

[14] 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.

[15] 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.

[16] Benjamin Glas, Jorge Guajardo, Hamit Hacioglu, Markus Ihle, Karsten Wehefritz, and Attila Yavuz. Signal-based automotive communication security and its interplay with safety requirements. In *Proceedings of Embedded Security in Cars Conference*. Citeseer, 2012.

[17] Mohamed Grissa, Attila A. Yavuz, and Bechir Hamdaoui. An efficient technique for protecting location privacy of cooperative spectrum sensing users. In *IEEE Infocom Green and Sustainable Networking and Computing Workshop, 2016 (GSNC '16)*., April 2016.

[18] 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.

[19] Mohammad Kamrul Hasan, AKM Ahasan Habib, Zarina Shukur, Fazil Ibrahim, Shayla Islam, and Md Abdur Razzaque. Review on cyber-physical and cyber-security system in smart grid: Standards, protocols, constraints, and recommendations. *Journal of Network and Computer Applications*, 209:103540, 2023.

[20] Andreas Hülsing, Denis Butin, Stefan Gazdag, Joost Rijneveld, and Aziz Mohaisen. Xmss: extended merkle signature scheme, 2018.

[21] 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.

[22] Matthias J Kannwischer, Joost Rijneveld, Peter Schwabe, and Ko Stoffelen. pqm4: Testing and benchmarking nist pqc on arm cortex-m4. 2019.

[23] V. Kounev, D. Tipper, Attila A. Yavuz, B.M. Grainger, and G.F. Reed. A secure communication architecture for distributed microgrid control. *Smart Grid, IEEE Transactions on*, PP(99):1–9, May 2015.

[24] 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.

[25] Jaeheung Lee and Yongsu Park. Horsic+: An efficient post-quantum few-time signature scheme. *Applied Sciences*, 11(16):7350, 2021.

[26] Lingyun Li, Xianhui Lu, and Kunpeng Wang. ebiba: A post-quantum hash-based signature with small signature size in the continuous communication of large-scale data. *The Computer Journal*, 2023.

[27] Jianyuan Lu, Tong Yang, Yi Wang, Huichen Dai, Linxiao Jin, Haoyu Song, and Bin Liu. One-hashing bloom filter. In *2015 IEEE 23rd international symposium on quality of service*. IEEE, 2015.

[28] 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.

[29] 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.

[30] 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.

[31] Josef Pieprzyk, Huaxiong Wang, and Chaoping Xing. Multiple-time signature schemes against adaptive chosen message attacks. In *Selected Areas in Cryptography: 10th Annual International Workshop, SAC*. Springer, 2004.

[32] Victoria Y Pillitteri and Tanya L Brewer. Guidelines for smart grid cybersecurity. 2014.

[33] Leonid Reyzin and Natan Reyzin. Better than biba: Short one-time signatures with fast signing and verifying. Springer, 2002.

[34] Masoumeh Shafieinejad and Reihaneh Safavi-Naini. A post-quantum one time signature using bloom filter. In *2017 15th Annual Conference on Privacy, Security and Trust (PST)*, pages 397–3972. IEEE, 2017.

[35] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Review*, 1999.

[36] 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.

[37] Zheng Yang, Chenglu Jin, Yangguang Tian, Junyu Lai, and Jianying Zhou. Lis: Lightweight signature schemes for continuous message authentication in cyber-physical systems. In *Proceedings of the ACM Asia Conference on Computer and Communications Security*, 2020.

[38] 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.

[39] 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.

[40] Gaofeng Zhao, Rui Liu, Yang Li, Jin Huang, Mingxuan Zhang, and Weiwei Miao. Multi-user broadcast authenticaiton in power lte private network with compressed bloom filter. In *2021 IEEE 5th International Conference on Cryptography, Security and Privacy*. IEEE, 2021.

[41] Fei Zhu, Xun Yi, Alsharif Abuadbba, Junwei Luo, Surya Nepal, and Xinyi Huang. Efficient hash-based redactable signature for smart grid applications. In *European Symposium on Research in Computer Security*. Springer, 2022.