Trustworthy and Efficient Digital Twins in Post-Quantum Era with Hybrid Hardware-Assisted Signatures

SAIF E. NOUMA and ATTILA A. YAVUZ, University of South Florida, USA

Digital Twins (DT) virtually model cyber-physical objects via sensory inputs by simulating or monitoring their behavior. Therefore, DTs usually harbor vast quantities of Internet of Things (IoT) components (e.g., sensors) that gather, process, and offload sensitive information (e.g., healthcare) to the cloud. It is imperative to ensure the trustworthiness of such sensitive information with long-term and compromise-resilient security guarantees. Digital signatures provide scalable authentication and integrity with non-repudiation and are vital tools for DTs. Post-quantum cryptography (PQC) and forward-secure signatures are two fundamental tools to offer long-term security and breach resiliency. However, NIST-PQC signature standards are exorbitantly costly for embedded DT components and are infeasible when forward-security is also considered. Moreover, NIST-PQC signatures do not admit aggregation, which is a highly desirable feature to mitigate the heavy storage and transmission burden in DTs. Finally, NIST recommends hybrid PQ solutions to enable cryptographic agility and transitional security. Yet, there is a significant gap in the state of the art in the achievement of all these advanced features simultaneously. Therefore, there is a significant need for lightweight digital signatures that offer compromise resiliency and compactness while permitting transitional security into the PQ era for DTs.

We create a series of highly lightweight digital signatures called Hardware-ASisted Efficient Signature (HASES) that meets the above requirements. The core of HASES is a hardware-assisted cryptographic commitment construct oracle (CCO) that permits verifiers to obtain expensive commitments without signer interaction. We created three HASES schemes: PQ−HASES is a forward-secure PQ signature, LA−HASES is an efficient aggregate Elliptic-Curve signature, and HY−HASES is a novel hybrid scheme that combines PQ−HASES and LA−HASES with novel strong nesting and sequential aggregation. HASES does not require a secure-hardware on the signer. We prove that HASES schemes are secure and implemented them on commodity hardware and and 8-bit AVR ATmega2560. Our experiments confirm that PQ−HASES and LA−HASES are two magnitudes of times more signer efficient than their PQ and conventional-secure counterparts, respectively. HY−HASES outperforms NIST PQC and conventional signature combinations, offering a standard-compliant transitional solution for emerging DTs. We open-source HASES schemes for public-testing and adaptation.

CCS Concepts: • **Security and Privacy** → **Cryptography**; • **Computer systems organization** → **Architectures**; • **Hardware** → **Emerging Technology**.

Additional Key Words and Phrases: Multimedia authentication, digital twins, post-quantum security

1 INTRODUCTION

Digital Twins (DT) paradigm aims to represent a digital replica of the physical systems [1]. It can facilitate the means to monitor, understand, and optimize the functions of physical entities living, or non-living [26]. They are primarily empowered by Internet of Things (IoT) components (e.g., smart sensors, actuators) to approximate the behavior of twins. Hence, DTs can play an important role in many real-life applications such as constructing parallel metaverse world. In particular, DTs are suitable for IoT-cloud enabled systems with human-centric applications

Authors' address: Saif E. Nouma, saifeddinenouma@usf.edu; Attila A. Yavuz, attilaayavuz@usf.edu, University of South Florida, 3720 Spectrum Blvdve, Interdisciplinary Research Building (IDR)-400, Tampa, Florida, USA, 33612.

ACM 1551-6857/2023/12-ART
https://doi.org/10.1145/3638250

[49]. For instance, IoT devices of a patient (e.g., medical implants, haptic sensors) constantly gather highly security sensitive information to be securely offloaded to a remote cloud/edge server for data analytics and smart decision-making [1]. Subsequently, such medical DT can aid physicians in monitoring the patient's well-being and contribute to the prevention of potential illnesses from manifesting or advancing.

DT applications gather, process and offload a large volume of heterogeneous data streams, which entail potentially security-sensitive information (e.g., healthcare, financial, personal). Therefore, it is imperative to ensure the trustworthiness of sensitive data maintained by the DT-Cloud continuum. Especially, authentication and integrity are foundational security measures for safeguarding DT systems against a range of potential attacks (e.g., data tampering, fraud, man-in-middle). Digital signatures [7] provide scalable authentication/integrity as well as non-repudiation and public verifiability via public-key infrastructures. Hence, they are essential primitives to ensure security and trust for DTs. Yet, DT applications have security and performance requirements that are well beyond what conventional (standard) signatures can offer. Below, we outline some of the important properties that a digital signature should offer to be a viable solution for emerging DTs.

(i) Post-Quantum (PQ) Security: With the anticipated arrival of quantum computers [15], Shor's algorithm [50] can break conventional cryptosystems (e.g., ECDSA [7], RSA [32]) that rely on conventional intractability assumptions (e.g., Discrete Logarithm Problem (DLP) [32]). The fast progress in quantum computing make it necessary to swiftly transition to quantum-secure alternatives. Indeed, NIST initiated PQC standardization, outlining selected PQ signature standards. The White House also issued a memorandum providing guidance for the transition to PQC [1]. Thus, DT applications urgently need a PQ digital signature that respects their efficiency requirements.

(ii) Compromise-resiliency: Most of the DT applications such as smart-health monitoring or smart-home, sensors, actuators, and mobile devices may operate in open and hostile environments that make them vulnerable to compromise via either physical means or malware [56]. Hence, it is important for a digital signature to offer compromise-resiliency features like forward security (FS) [48]. The latter prevents forgeries for the past time periods upon private key exposure events. Hence, it guarantees authenticity and integrity before a system breach point [35] by constantly updating the secret keys. FS signatures are (in some cases significantly) costlier than their standard (plain) signature counterparts. This becomes even more expensive when PQ security is also considered.

(iii) Signer Efficiency: DT systems harbor large quantities of low-end IoT devices (e.g., smart sensors) that are highly resource-limited (e.g., battery, CPU, memory, bandwidth). Hence, it is crucial to minimize the overhead of digital signatures over DT systems to permit a secure yet practical deployment [29]. Hence, the digital signature should offer computationally lightweight operations and small signature sizes to reduce the impact of cryptographic overhead on battery life. Yet, even conventional signature standards (e.g., ECDSA) are considered expensive for some IoT applications. More importantly, the NIST PQC signatures are infeasible on low-end platforms, and currently impractical with advanced features like forward security.

(iv) Ease of Transitioning and Cryptographic Agility: The transition to PQC is a challenging and error-prone progress due to the need to update current cryptographic implementations. Several factors need to be considered, including cryptographic solutions for heterogeneous hardware devices (e.g., IoT devices). Thus, it is highly desirable to adopt PQC that offers backward compatibility with prior implementations, which can facilitate the implementation and transition process. For instance, hash-based schemes utilize cryptographic hash functions (e.g., SHA-2), which are already present in all standard-compliant cryptographic software libraries. Moreover, hash-based algorithms are considered PQ-safe with minimal intractability assumptions.

NIST emphasizes the need for a *hybrid PQ signature* that combines both classical and post-quantum variants [5]. The hybrid signature offers the advantage that as long as at least one of the algorithms used remains secure, the hybrid scheme will remain secure [9]. It not only enhances security definitions by requiring adversaries to break each algorithm but also allows for the utilization of existing libraries of traditional cryptosystems while fulfilling

[1] https://www.whitehouse.gov/wp-content/uploads/2022/11/M-23-02-M-Memo-on-Migrating-to-Post-Quantum-Cryptography.pdf

some PQC. This concept promotes *cryptographic agility* [31], which involves designing protocols that can support multiple algorithms simultaneously, as a safety and security measure against the vulnerability of deployed ones.

1.1 Related Work and Limitations of the State-of-the-Art

We focus on digital signatures with desirable properties for DT applications. First, we discuss special signature schemes and their constraints, followed by other complementary related works.

Digital Signatures with Special Properties. We discuss the most relevant signatures to ours with an emphasis on PQC, compromise-resiliency, compactness (aggregate) and finally hybrid features.

• *Post-Quantum (PQ) Digital Signatures:* As part of the NIST PQC standardization efforts [37], the selected PQ signature standards include two lattice-based schemes, Dilithium [24] and Falcon [28], and a hash-based SPHINCS+ [8]. SPHINCS+ incurs high energy and bandwidth overheads due to expensive signature generation and large key sizes, making it infeasible for frequent data authentication on low-end devices. The two lattice-based signatures offer a better balance between the performance metrics (i.e., storage, transmission, computation), but are based on new intractability assumptions, compared to hash-based schemes. To our knowledge, the implementation of NIST standards still does not support the resource-constrained devices (e.g., 8-bit microcontrollers (MCUs)) which are widely deployed in the IoTs. For example, implementing Falcon [61] on such devices is challenging as it requires double-precision floating-point operations, which are not supported by many low-end devices. The lattice-based BLISS is the only PQ signature with benchmarking on an 8-bit device but it is vulnerable to numerous side-channel and timing attacks [27].

• *Forward-Secure (FS) Digital Signatures:* offer improved protection against system breaches (e.g., malware attacks) by employing a key update strategy [16]. The frequent private key update results in increased size of public keys. A straightforward technique consists of refreshing the private key via a one-way hash function which results in a linear public key blow-up w.r.t. the number of messages to be signed [59]. This results in a linear bandwidth overhead at the signer which is deemed to be a resource-constrained device. Thus, it is not scalable to the large DT environments. On the other side, there exist generic transformations (e.g., [47, 58]) that introduce forward security property to a normal digital signature. The most efficient approach introduces a $log_2(J)$ signing overhead blow-up, where J is the maximum number of messages to be signed. The extra overhead exacerbates when PQ security is considered. For example, the RFC standard and NIST recommendation, XMSS^{MT} [18] is currently the only PQ and FS signature standard. Its signature generation is more than magnitudes times costlier than the NIST PQC standard Dilithium. Another line of research (e.g., a lattice-based FS-PQ digital signature ANT [6]) enables signers to delegate the public-key computation to a set of distributed servers. Despite their efficient signing and compact key sizes, such approaches assume non-colluding servers that not only risky for some DT applications but also introduces heavy network delays. Additionally, there exist a recent identity-based digital signature with forward security [48]. However, it does not offer a lightweight signing and no benchmark on low-end device is reported.

• *Aggregate Digital Signatures:* Aggregate Signature (AS) are vital for frequent and continuous authentication, particularly on low-end devices and bandwidth-limited networks. Formally, an AS scheme reduces the cryptographic overhead by combining multiple distinct signatures, that can be issued from a single or multiple signers, into a fixed-length signature. Thus, aggregation improve the performance efficiency by lowering the cryptographic payload into a constant overhead. ASs have seen numerous applications, namely in IoT networks [39], delay-tolerant networks [62], secure routing [10], and distributed systems (e.g., Blockchain) [23].

Several aggregate signatures have been proposed in the literature which can be divided into: *(i) Factorization-based:* rely on the hardness assumption of the factorization problem. For example, the condensed RSA (C-RSA) [57] signature offers aggregation with an efficient verification but expensive signing and large key sizes due to the modular exponentiation over large prime modulus. *(ii) Pairing-based:* based on bilinear maps and offer aggregation within multi-user settings. BLS [13] represents the widely-used seminal pairing-based signature which

offers a small signature and public key sizes, but with a costly pairing and map-to-point operations that are highly expensive. Recent optimized pairing-based aggregate signatures are back-traced to BLS in terms of signer efficiency. For instance, LFS-AS [16] is a pairing-based aggregate signature with FS for e-Health. Its signature generation performs four exponentiation which are expensive for 8-bit devices. Its verification is even more costlier since due to pairing operations. Numerous aggregate signatures with advanced properties (e.g., certificateless setting) [52] are pairing-free schemes. However, such recent works focus on reducing the verification computational cost while omitting the prohibitive signing overhead for 8-bit MCUs. *(iii)* Elliptic Curve (EC)-based: The aggregate EC-based schemes offer a balance in terms of signature generation efficiency and key sizes. For instance, FI-BAF [59] is a signer-efficient EC-based AS and FS digital signature. However, it incurs persistent linear storage overhead at verifiers, wherein one-time public keys must be periodically redistributed. Thus, it is not scalable for large-scale data-intensive DT applications. There exist recent EC-based AS schemes with advanced properties for IoTs (e.g., [34, 52]). However, they inherit similar signing efficiency to that of Schnorr [20] at the signer side and they lack low-level implementation on resource-constrained devices.

There is very limited work on PQ aggregate signatures. For example, a recent lattice-based AS scheme [12] only works for non-interactive one-time or interactive multiple-time settings, with a logarithmic compression rate w.r.t the number of signatures. Another approach converts classical DLP-based aggregate signatures into lattice dimensions using the Fiat-Shamir with Aborts paradigm [14], but this method requires additional computational overhead and again very low compression ratios. Additionally, there exist numerous lattice-based aggregate signatures, optimized for blockchain applications (e.g., [4]). Again, they have costly signature generation with the absence of benchmark on low-end IoT. Therefore, PQ-secure ASs are currently not feasible for our envisioned applications.

• *Hybrid Approaches:* Various hybrid digital signatures aim to establish robust authentication methods by combining different digital signatures and other emerging technologies (e.g., quantum networks). For instance, [61] proposed a hybrid protocol that incorporates NIST PQC standards, quantum networks, and hardware acceleration methods, thus enabling swift and quantum-safe execution of distributed protocols. There exist hybrid key agreement protocols (e.g., [45]) that employ post-quantum cryptography and physical layer methods, which are orthogonal to ours. Crocket et al. [21] integrate PQ and hybrid digital signatures into Internet security protocols (e.g., TLS). Those works are complementary to ours. Paul et al. [43] benchmark a range of hybrid digital signatures by combining NIST standards including conventional (e.g., RSA) and PQ (e.g., Falcon) signature schemes. However, our findings reveal that both categories involve costly signing operations, making them unsuitable for resource-limited devices even before considering advanced security properties. Notably, there is a gap in hybrid signature solutions that effectively address quantum threats while also embodying the desirable attributes of conventional signatures. NIST underscores the significance of hybrid conventional-PQ-secure cryptosystems [5] to offer fail-safe designs against unexpected failure of emerging PQC schemes [9, 40], while also retaining cryptgoraphic agility [31].

Other/Complementary Related Work.

• *Secure Hardware-Assisted Primitives*: Another line of research exploits the availability of trusted execution environments in modern architectures to achieve higher cryptographic functionalities. For instance, it is possible to emulate asymmetric cryptosystems from symmetric-key algorithms (e.g., MACs) with a secure enclave (e.g., Intel Software Guard (SGX) [25]). While efficient and foreseen to be PQ-secure, it requires each party to have a local secure enclave (e.g., SCB [41]), which is not practical for low-end devices that represent a major part of DTs. Moreover, they provide restricted public verifiability (only SGX-enabled devices) and lack non-repudiation (due to shared symmetric keys), which is a critical need for numerous real-life applications.

• *Other Special Authentication Techniques:* Proof of Data Possession (PDP) [3] and Proof of Retrievability (PoR) [2] protocols can offer public auditing of the outsourced user data. They offer fast verification time of the authenticated data via interactive random checks. They mostly rely on homomorphic linear authenticators (HLA) [53] which allows auditors to perform verification without retrieving the entire data. Most HLAs are also implemented via the pairing-based or RSA-based schemes. Despite their merits, most of the above approaches rely

on foundational operations/signatures such as EC scalar multiplications (e.g., Ed25519), RSA, or BLS signatures at the signer. Moreover, we observe the absence of performance evaluations on low-end devices (e.g., 8-bit ATMega2560). In our comparisons, we focus on Ed25519 [7], RSA [32], and BLS [13] to represent the signer overhead of the schemes that rely on such operations (see Section 6).

1.2 Our Contribution

We created a new series of highly lightweight digital signatures called *HArdware-Assisted Efficient Signatures (HASES)*. HASES efficiently combines above seemingly conflicting attributes while ensuring near-optimal signer performance to meet resource-constrained DT requirements. Our main component is a hardware-assisted cryptographic commitment construct oracle (CCO) that supplies verifiers with expensive one-time commitments without signer interaction. We realized CCO via secure enclaves and created three HASES instantiations: *(i) Post-Quantum PQ-HASES:* achieves post-quantum security and forward security simultaneously. PQ-HASES transforms the one-time hash-based HORS [46] into multiple-time without consorting with heavy sub-tree construction or secure enclaves on signers. *(ii) Lightweight LA-HASES:* is a conventional-secure partially aggregate signature based on Curve25519 on which the standard Ed25519 operates. LA-HASES avoid running expensive EC scalar multiplication or pairing operations on signers by harnessing CCO as the supplier of the costly EC commitments. *(iii) Hybrid HY-HASES:* combines LA-HASES and PQ-HASES via a novel nesting approach. This achieves partial aggregation, reinforced by a PQ-FS umbrella signature. Moreover, HY-HASES solely employs standard-compliant operations, ensuring backward compatibility. Unlike previous approaches, HASES are the first, to the best of our knowledge, to provide a conventional-PQ hybrid signature that adheres to NIST standards without incurring heavy overhead or relying on secure hardware on signers.

Significant Improvements over Preliminary Version and Prior Works: A small part of this paper, specifically the PQ-HASES scheme, accepted in [38]. Our current article makes substantial new contributions over its preliminary version on algorithmic novelty, formal security analysis, and experiment/implementation fronts (more than 18 pages of this 28-page manuscript are new content):

(1) Introducing LA-HASES as non-interactive signer-optimal aggregation via CCO: Hash-based digital signatures (e.g., PQ-HASES) lack aggregation due to their non-algebraic structure. Most recent lightweight digital signatures (e.g., [52]) are instantiated from the primitive BLS [52] which lack efficient signature generation. EC-based signatures (e.g., Schnorr [20]) offer improved efficiency but still require expensive commitment generation on low-end signers, which results into a linear storage and bandwidth overhead. Numerous Schnorr-based aggregate signatures (e.g., [17]) offer efficient verification but incur interaction between signers, thus unpractical for IoT. Various techniques (e.g., [39, 42, 59]) have been developed to mitigate this bottleneck. For example, FI-BAF [59] involves precomputing commitments during key generation and storing them on verifiers beforehand. Yet, this approach is impractical for large DT-enabled IoT networks due to its linear verifier storage overhead per user. ESEM [42] separates commitment generation from signing and delegates it to a set of distributed servers. While this enhances signing speedup, it becomes susceptible to networks delays and non-colluding server assumptions. To the best of our knowledge, LA-HASES is the first to incorporate secure enclaves for commitment generation. This allows for non-interactive and efficient aggregate signing with minimal storage overhead for verifiers. Verifiers can flexibly request one-time commitments in offline mode or on-demand. LA-HASES also offers improved signing efficiency and produce aggregate signatures that are 10× smaller than the initial PQ-HASES.

(2) Introducing HY-HASES via a Novel Strong Nesting Strategy: Our initial PQ-HASES offers high signing efficiency and provides both post-quantum and forward security. However, it lacks aggregation which is crucial for reducing bandwidth and battery usage on low-end devices. On the other hand, LA-HASES efficiently aggregates signatures and requires minimal cryptographic storage for verifiers. However, it falls short of providing post-quantum security, which is essential for long-term security in distributed DT applications. Note that both

PQ−HASES and LA−HASES adhere to standard compliance and backward compatibility. As discussed in Section 1.1, previous hybrid signatures constructions have not investigate the combinations of digital signatures with different security features (e.g., FS, aggregation). To the best of our knowledge, HY−HASES is the first hardware-assisted solution that offers signer-optimal hybrid capabilities, achieving aggregation, forward security, and PQ promise simultaneously. This is achieved through introducing a novel nesting method that uniquely combines LA−HASES and PQ−HASES digital signatures.

(3) Expanding Performance Analysis with Further Optimizations: We fully implemented all HASES schemes on both commodity hadware and IoT devices and tested them on actual DT datasets. We added new experiments and comparisons with the state-of-the-art PQ, aggregate, and hybrid schemes. Our experiment results showcases a speedup over initial version thanks to the use of NIST standard Ascon [22] for lightweight hashing and key derivation.

Desirable Properties of HASES Over Prior Works: We outline desirable properties as follows:

(1) Signer Computation and Energy Efficiency: PQ−HASES needs only a small-constant number of hash calls (i.e., ≈ 20) per FS signing making it 1542× and 16.28× faster than XMSS^{MT} and Dilithium, which is the only FS and the most signer efficient NIST PQC standard (not FS), respectively. PQ−HASES is 9.34× faster than Ed25519 which is neither PQ nor FS. LA−HASES also runs only a few hash calls and modular arithmetic operations whose costs are negligible. This makes it 762× and 183× faster than BLS and C-RSA, respectively. PQ−HASES and LA−HASES are 40× and 99× more energy efficient than BLISS and Ed25519 which are the only feasible PQ and conventional-secure alternatives on the 8-bit device, respectively. The HY−HASES is several magnitudes more efficient in signing with smaller tag sizes compared to alternative combinations (e.g., XMSS^{MT}-C-RSA, Dilithium-BLS), while also offering stateful signing and backward compatibility.

(2) Compact Signatures: The signature size of PQ−HASES is identical to HORS, making it the most compact signature among its PQ counterparts (e.g., 10× smaller than only PQ-FS alternative XMSS^{MT}). The signature size of LA−HASES is comparable to that of the standard non-aggregate Ed25519 and the short signature BLS, but without expensive EC scalar multiplication and map-to-point operations, respectively. HY−HASES offers the smallest AS and FS signature with PQ security, which is 44× smaller than the most compact Dilithium-BLS combination.

(3) Non-Interactive and Scalable Multi-Users: HASES schemes lifts the burden of conveying/certifying one-time commitments from the signer, thereby making it independent from CCO and verifiers. CCO can supply any public commitment of a valid signer to verifiers either beforehand or on-demand, with adjustable storage overhead, permitting a scalable management service for massive-size DT-enabled IoT networks with millions of users.

(4) Architectural Feasibility and Cryptographic Agility: (i) Unlike some hardware-based solutions (e.g., [11, 41]), HASES does not require a secure enclave on the signer, which is not feasible for low-end IoT. (ii) HY−HASES is the first, to the best of our knowledge, to offer a hybrid conventional-PQ signature with partial aggregation and FS by only using standard primitives (i.e., SHA-256, Curve25519). Thus, it can expedite the PQC transition for DTs with its backward compatibility.

(5) High Security: (i) PQ−HASES only relies on SHA-256, and therefore is free from rejection/Gaussian sampling that permits devastating side-channel attacks in its lattice-based counterparts (e.g., BLISS-I [27]). (ii) The stateful signing of HASES avoids vulnerabilities of weak pseudo-random generators typically found in low-end IoT devices. (iii) PQ−HASES achieve both PQ and FS properties for long-term security guarantees. (iv) The signature aggregation offers resiliency against truncation attacks due to the all-or-nothing feature.

(6) Full-fledge Implementation: We fully implemented all HASES schemes on 8-bit MCU and commodity hardware at signers. For verifier side, we used a commodity hardware equipped with Intel SGX as the CCO. Our implementation can be found at: https://github.com/SaifNOUMA/HYHASES

2 PRELIMINARIES

The acrynoms and notations, used in the paper, are described in Table 1.

Table 1. List of notations and acronyms

Notation	Description	Acronym	Description
$\|\|$	string concatenation	IoT	Internet of Things
$\|x\|$	bit length of variable x	MCU	Microcontroller Unit
$x \xleftarrow{\$} \mathcal{S}$	random selection from a set $\mathcal{S}$	DT	Digital Twins
$\{0,1\}^*$	set of binary strings of any finite length	NIST	National Institute of Standards and Technology
$\{q_i\}_{i=0}^{n-1}$	set of items q_i for $i = 0, \ldots, n-1$	PQC	Post-Quantum Cryptography
$f : \{0,1\}^* \rightarrow \{0,1\}^\kappa$	one-way function	PQ	Post-Quantum security
$\mathtt{PRF} : \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}^\kappa$	key derivation function accepts a key and a message as input	sk/PK	Secret/Public keys
$H : \{0,1\}^* \rightarrow \{0,1\}^\kappa$	cryptographic hash function	FS	Forward Security
$H^{(\ell)}(.)$	ℓ consecutive hash evaluations	AS	Aggregate Signature
Epoch (or Batch) j	set of finite data recordings	EC	Elliptic Curves
x_j	variable during epoch j	PRF	Pseudo-Random Function
x_j^ℓ	variable during epoch j in the iteration ℓ	ROM	Random Oracle Model
$x_j^{\ell_1,\ell_2}$	aggregate variable during ℓ_1 and ℓ_2 iterations of epoch j	CCO	Commitment Construct Oracle

Definition 2.1. A signature scheme SGN is a tuple of three algorithms (Kg, Sig, Ver):

- $(sk, PK) \leftarrow \mathtt{SGN.Kg}(1^\kappa)$: Given the security level κ, it returns a private/public key pair (sk, PK).
- $\sigma \leftarrow \mathtt{SGN.Sig}(sk, M)$: Given sk and a message M, the signing algorithm returns a signature σ.
- $b \leftarrow \mathtt{SGN.Ver}(PK, M, \sigma)$: Given the public key PK, message M, and its corresponding signature σ, the verification algorithm outputs a bit b (if $b = 1$, the signature is valid, otherwise it is invalid).

Aggregate signature is as Def. 2.1 except signature generation/verification accept a set of messages.

Definition 2.2. A single-signer aggregate signature scheme ASGN is a tuple of four algorithms (Kg, ASig, Agg, AVer) defined as follows:

- $(sk, PK) \leftarrow \mathtt{ASGN.Kg}(1^\kappa)$: Given security parameter κ, it returns private/public keys (sk, PK).
- $\sigma_{1,L} \leftarrow \mathtt{ASGN.ASig}(sk, \vec{M})$: Given sk and a set of messages $\vec{M} = \{m_\ell\}_{\ell=1}^L$, it returns an aggregate signature $\sigma_{1,L}$.
- $\sigma_{1,L} \leftarrow \mathtt{ASGN.Agg}(\{\sigma_\ell\}_{\ell=1}^L)$: Given L distinct signatures $\{\sigma_\ell\}_{\ell=1}^L$, it returns aggregate tag $\sigma_{1,L}$.
- $b \leftarrow \mathtt{ASGN.AVer}(PK, \vec{M}, \sigma_{1,L})$: Given PK, a set of messages $\vec{M} = \{m_\ell\}_{\ell=1}^L$, and its aggregate signature $\sigma_{1,L}$, it outputs a bit b (if $b = 1$, the signature is valid, otherwise invalid).

Forward security [35] periodically evolves the private key and deletes its previous iterations, thereby enhances breach resiliency against key compromise attacks.

Definition 2.3. A FS signature FSGN have four algorithms (Kg, Sig, Upd, Ver) defined below:

- $(sk, PK) \leftarrow \mathtt{FSGN.Kg}(1^\kappa, J)$: Given the security parameter κ and the maximum number of key updates J, it returns a private/public key pair (sk, PK).
- $sk_{j+1} \leftarrow \mathtt{FSGN.Upd}(sk_j, J)$: Given the private key sk_j, it returns sk_{j+1} and delete sk_j, if $j < J$, else aborts.
- $\sigma_j \leftarrow \mathtt{FSGN.Sig}(sk_j, M_j)$: Given sk_j and a message M_j, it returns a FS signature σ_j as output and perform key update $sk_{j+1} \leftarrow \mathtt{FSGN.Upd}(sk_j, J)$, if $j < J$. Otherwise, it aborts.
- $b \leftarrow \mathtt{FSGN.Ver}(PK, M_j, \sigma_j)$: Given PK, a message M_j, and its corresponding signature σ_j, the verification algorithm outputs a validation bit b (if $b = 1$, signature is valid, otherwise invalid).

A hybrid signature scheme consists of a combination of two (or more) distinct digital signatures. The resulting hybrid scheme is unforgeable as long as at least one of the underlying schemes remains secure. An example is

the fusion of conventional and PQ signature schemes. NIST recommends hybrid schemes due to their enhanced security against unexpected algorithmic breaches and ease of transition [5]. Of particular interest is the nested combination strategy [9], as defined below:

Definition 2.4. A hybrid signature scheme (HYSGN) is a composition of two distinct signature schemes $\texttt{SGN}_1$ and $\texttt{SGN}_2$. It is described as follows:

- $(sk, PK) \leftarrow \texttt{HYSGN.Kg}(1^\kappa)$: Given the security parameter κ, it generates $(sk_1, PK_1) \leftarrow \texttt{SGN}_1\texttt{.Kg}(1^\kappa)$ and $(sk_2, PK_2) \leftarrow \texttt{SGN}_2\texttt{.Kg}(\kappa)$. It returns $(sk \leftarrow \langle sk_1, sk_2\rangle, PK \leftarrow \langle PK_1, PK_2\rangle)$.
- $\sigma \leftarrow \texttt{HYSGN.Sig}(sk, M)$: Given the private key sk and the message M, it computes $\sigma_1 \leftarrow \texttt{SGN}_1(sk_1, M)$ and $\sigma_2 \leftarrow \texttt{SGN}_2(sk_2, M\|\sigma_1)$. It returns $\sigma \leftarrow \langle \sigma_1, \sigma_2\rangle$.
- $b \leftarrow \texttt{HYSGN.Ver}(PK, M, \sigma)$: Given PK, the message M, and the signature σ, it computes $b_1 \leftarrow \texttt{SGN}_1\texttt{.Ver}(PK_1, M, \sigma_1)$ and $b_2 \leftarrow \texttt{SGN}_2\texttt{.Ver}(PK_2, M, \sigma_2)$. It returns $(b \leftarrow b_1 \wedge b_2)$.

A hardware-assisted digital signature removes the need of supplying commitments and public keys (with certificates) from signers. It introduces a third-party entity, *Commitment Construct Oracle (CCO)*, equipped with a secure enclave (e.g., Intel SGX [25]) that issues cryptographic keys to verifiers either offline or on-demand. A PQ CCO can be achieved by using PQ cryptographic operations within the enclave [11]. A hardware-assisted digital signature can be extended into a multi-user setting by deriving the users' private keys from a master key, which is solely stored at CCO. We call such scheme as Hardware-Assisted Mutli-User signature (HAMU−SGN), described in Def. 2.5.

Definition 2.5. A Hardware-Assisted Multi-User signature scheme HAMU−SGN consists of four algorithms (Kg, ComConstr, Sig, Ver) defined as follows:

- $(msk, \vec{sk}, I) \leftarrow \texttt{HAMU-SGN.Kg}(1^\kappa, \vec{ID}, J)$: Given the security level κ, the signers' identities $\vec{ID}$ and the max number of signatures J, it returns a master key msk, users' private keys $\vec{sk}$, and the system parameters I.
- $\sigma_i^j \leftarrow \texttt{HAMU-SGN.Sig}(sk_i, M_i^j)$: Given the private key sk_i of the user ID_i and the message M_i^j, it returns the signature σ_i^j, given that the counter $j \leq J$. It updates $j \leftarrow j + 1$.
- $C_i^j \leftarrow \texttt{HAMU-SGN.ComConstr}(msk, ID_i, j)$: Given the master key msk, the signer identity $ID_i \in \overrightarrow{ID}$, and the state $j \leq J$, it returns the corresponding commitment C_i^j under msk.
- $b_i^j \leftarrow \texttt{HAMU-SGN.Ver}(\langle PK_i^j, C_i^j\rangle, M_i^j, \sigma_i^j)$: Given PK_i^j, the commitment C_i^j, a message M_i^j, and its associated signature σ_i^j, it returns a bit b_i^j, with $b_i^j = 1$ meaning *valid*, and $b_i^j = 0$ otherwise.

Our HAMU−SGN instantiations: can be classified to several types of signature schemes as folllows:

- *Aggregate-based signatures:* The signature generation and verification algorithms of the aggregate HAMU−SGN (HAMU−ASGN) follows Def. 2.2. Hence, HAMU−ASGN follows the A−EU−CMA security model (see Def. 3.2).
- *Forward-secure signatures (see Def. 2.3):* A commitment construct algorithm (ComConstr) returns one-time commitments based on the private key updates at the signer. Thus, the forward-secure HAMU−SGN (HAMU−FSGN) follows the F−EU−CMA security model (see Def. 3.1).
- *Hybrid signatures (see Def. 2.4)*: A hybrid HAMU−SGN(HYHAMU−SGN) can be constructed from various HAMU−SGN instantiations. In section 4, we discuss a HYHAMU−SGN signature, built from a fusion of an aggregate HAMU−ASGN and a FS-PQ HAMU−FSGN digital signatures.

3 MODELS

3.1 System Model

Our system model is suitable for a DT-enabled IoT network, wherein a large number of low-end devices (e.g., sensors) and various metaverse devices (e.g., camera, AR headset, smart glasses) authenticate their generated data (e.g., images, haptic biometrics, Electrocardiogram (ECG)-signals) and offload them to a remote edge server. Our model consists of three entities:

(i) Signers: In a DT-enabled metaverse framework, various low-end devices (e.g., implants, sensors) and metaverse equipment (e.g., camera, headset) generate continuous multimedia data (e.g., haptic, image). This data requires offloading to a remote edge cloud for future analytics and decision-making. Thus, optimizing signautre generation and cryptographic payload is vital to prolong battery life. The low-end devices are also expected to operate in adversarial environments, under various attacks (e.g., key compromise [59], quantum [31], truncation and delayed [59] attacks). Hence, an ideal authentication scheme should provide FS for breach resilience, PQ security against quantum attacks, and aggregation to protect against truncation attacks while supporting high transmission rates. As the signers are low-end devices, secure enclaves are not required at their side. Hence, we consider them broadcasting authenicated data without involving third parties or conveying public keys to verifiers.

(ii) Verifier: encompass any untrusted entities (laptop, cloud server). They receive signatures from signers and one-time keys from a third-party supplier, either on-demand or offline. Verifiers are unconstrained by resource limitations and can possess a secure hardware. In our system model, the verifier represents the edge server which is responsible of receiving authenticated data from signers and conducting data processing for insightful analytics and smart decision making. Therefore, we justify the use of secure enclaves on verifiers to exclusively store the user keys. Meanwhile, our system model enables any other entity to perform verifications.

(iii) Commitment Construct Oracle (`CCO`): We introduce a commitment constructor that supplies verifiers with one-time keys. Digital signatures featuring advanced security properties (e.g., FS, PQ security) incur linear bandwidth overhead in relation to transmitted data. This incur additional penalties on signers, rendering deployment nearly infeasible. This could drain significant energy, especially for the low-end signers. Moreover, It will add up a high bandwidth overhead and incur traffic congestion across public networks. The `CCO` role is relieve signer from the burden of certifying one-time keys. `CCO` can live on verifiers to minimize communication delays in on-demand requests. We implemented `CCO` with Intel SGX which is widely available on cloud plateforms [25]. However, `HASES` can be realized with *any* secure hardware offering a trusted execution environment.

3.2 Threat and Security Model

We define the standard *Forward-secure Existential Unforgeability against Chosen Message Attack* (`F-EU-CMA`) [35] by incorporating `CCO`. It serves as a threat model of `HAMU-FSGN`. The adversary $\mathcal{A}$ obtain J FS-signatures under a challenge PK. A `HAMU-FSGN` scheme is proven to be `F-EU-CMA` based on the experiment in Def. 3.1, wherein $\mathcal{A}$ is given four oracles defined as follows:

(i) *Random Oracle RO(.):* $\mathcal{A}$ is given access to a random oracle from which it can request the hash of any message M_j of her choice, up to q_s' messages. Note that in our proofs (see Appendix B), cryptographic hash functions $H_{k=0,1,2}$ are modeled as random oracles via $RO(.)$ (Random Oracle Model (ROM)) [32]. We note that some of our schemes explicitly use $RO(.)$, while others do not query $RO(.)$ but the security of their base signature scheme is secure in ROM (so as ours).

(ii) *Signing oracle $\texttt{Sig}_{sk}(.)$:* $\mathcal{A}$ is provided with a signing oracle $\texttt{Sig}_{sk}(.)$ on a message M_j, of her choice. $\mathcal{A}$ can query $\texttt{Sig}_{sk}(.)$ up to q_s individual messages in total as described, until she decides to "break-in".

(iii) *Break-in Oracle `Break-In(j)`:* $\mathcal{A}$ is provided with break-in oracle `Break-In(j)` which provides the current private key sk_j. Formally, if $\mathcal{A}$ queried $j < J$ individual messages to $\texttt{Sig}_{sk}(.)$, then the

`Break-In(j)` oracle returns the $(j+1)^{\text{th}}$ private key sk_{j+1} to $\mathcal{A}$. Otherwise, if $j \geq J$, then the break-in oracle rejects the query, as all private keys were used.

(iv) *Commitment Construct Oracle* $\texttt{ComConstr}_{msk}(.)$*:* $\mathcal{A}$ is provided with commitment construct oracle $\texttt{ComConstr}_{msk}(.)$ on a public commitment of her choice. We note that this commitment may either be a component of the public key or serve as the public key itself.

Definition 3.1. `F-EU-CMA` experiment $Expt^{\texttt{F-EU-CMA}}_{\texttt{HAMU-FSGN}}$ is as:
Experiment $Expt^{\texttt{F-EU-CMA}}_{\texttt{HAMU-FSGN}}(\mathcal{A})$:

$(msk, \overrightarrow{sk}, I) \leftarrow \texttt{HAMU-FSGN.Kg}(1^{\kappa})$
$(M^*, \sigma^*) \leftarrow \mathcal{A}^{RO(.),\ \texttt{HAMU-FSGN.Sig}_{sk}(.),\ \texttt{HAMU-FSGN.ComConstr}_{msk}(.),\texttt{Break-In(j)}}(.)$
If $\texttt{HAMU-FSGN.Ver}(\langle PK, C^*\rangle, M^*, \sigma^*) = 1$ and M^* was not queried to $\texttt{Sig}_{sk}(.)$ where C^* is the output of $\texttt{ComConstr}_{msk}(.)$, return 1, else, return 0.

The `F-EU-CMA` advantage of $\mathcal{A}$ is $Adv^{\texttt{F-EU-CMA}}_{\texttt{HAMU-FSGN}}(t, q_s, q'_s, 1) = Pr[Expt^{\texttt{F-EU-CMA}}_{\texttt{HAMU-FSGN}}(\mathcal{A})] = 1]$,where $\mathcal{A}$ have time complexity t, making at most q'_s queries to $RO(.)$, q_s queries to $\texttt{HAMU-FSGN.Sig}_{sk}(.)$ and $\texttt{HAMU-FSGN.ComConstr}_{msk}(.)$ combined, and one query to `Break-In(j)`.

We define the *Aggregate Existential Unforgeability Under Chosen Message Attack* (`A-EU-CMA`) [13] security model that captures the threat model of a `HAMU-ASGN` scheme (in single-signer setting).

Definition 3.2. `A-EU-CMA` experiment $Expt^{\texttt{A-EU-CMA}}_{\texttt{HAMU-ASGN}}$ is as:
Experiment $Expt^{\texttt{A-EU-CMA}}_{\texttt{HAMU-ASGN}}(\mathcal{A})$:

$(msk, \vec{sk}, I) \leftarrow \texttt{HAMU-ASGN.Kg}(1^{\kappa})$,
$(\vec{M}^*, \sigma^*) \leftarrow \mathcal{A}^{RO(.),\ \texttt{HAMU-ASGN.Sig}_{sk}(.),\texttt{ComConstr}_{msk}(.)}(PK)$,
If $1 = \texttt{HAMU-ASGN.Ver}(PK^*, \vec{M}^*, \sigma)$ and $\vec{M}^*$ was not queried to $\texttt{HAMU-ASGN.ASig}_{sk}(.)$, where PK^* was queried to `ComConstr`, return 1 else 0.

The `A-EU-CMA` of $\mathcal{A}$ is as $Adv^{\texttt{A-EU-CMA}}_{\texttt{HAMU-ASGN}}(t, q_s, q'_s) = Pr[Expt^{\texttt{A-EU-CMA}}_{\texttt{HAMU-ASGN}}(\mathcal{A}) = 1]$.
The `A-EU-CMA` advantage of `HAMU-ASGN` is as $Adv^{\texttt{A-EU-CMA}}_{\texttt{HAMU-ASGN}}(t, q_s, q'_s) = \max_{\mathcal{A}}\{Adv^{\texttt{A-EU-CMA}}_{\texttt{HAMU-ASGN}}(\mathcal{A})\}$, where the max is over $\mathcal{A}$ having time complexity t, with at most q'_s queries to $RO(.)$ and q_s queries to $\texttt{HAMU-ASGN.ASig}(.)$.

We define the *Hybrid Existential Unforgeability Under Chosen Message Attack* (`H-EU-CMA`) security model that captures the security model of a hybrid signature. We adopt the strong nesting technique as in [9]. The hybrid scheme `HYSGN` is constructed from two distinct signature schemes $\texttt{SGN}_1$ and $\texttt{SGN}_2$. Specifically, if either $\texttt{SGN}_1$ or $\texttt{SGN}_2$ are unforgeable in the classical (or quantum) random oracle model, then `HYSGN` is unforgeable in the classical (or quantum) random oracle model.

Definition 3.3. The `H-EU-CMA` experiment $Expt^{\texttt{H-EU-CMA}}_{\texttt{HYSGN}}$ is described as follows:
$(sk, PK) \leftarrow \texttt{HYSGN.Kg}(1^{\kappa})$, where $sk = \langle sk_1, sk_2\rangle$ and $PK = \langle PK_1, PK_2\rangle$,
$(M^*, \sigma^*) \leftarrow \mathcal{A}^{RO(.),\texttt{SGN.Sig}_{sk_1}(.),\texttt{SGN.Sig}_{sk_2}(.)}(PK)$, where $\sigma = \langle \sigma_1, \sigma_2\rangle$,
If $1 = \texttt{SGN}_1\texttt{.Ver}(PK_1^*, M^*, \sigma_1^*)$, $1 = \texttt{SGN}_2\texttt{.Ver}(PK_2^*, M^*\|\sigma_1, \sigma_2^*)$, and M^* was not queried to $\texttt{SGN}_1\texttt{.Sig}_{sk}(.)$, and $M^*\|\sigma_1$ was not queried to $\texttt{SGN}_2\texttt{.Sig}_{sk}(.)$, return 1, otherwise return 0.

The `EU-CMA` of $\mathcal{A}$ is defined as

$$Adv^{\texttt{H-EU-CMA}}_{\texttt{HYSGN}}(t, q_s, q'_s) = Pr[Expt^{\texttt{H-EU-CMA}}_{\texttt{HYSGN}}(\mathcal{A}) = 1] = Pr[Expt^{\texttt{EU-CMA}}_{\texttt{SGN}_1}(\mathcal{A}) = 1] \cdot Pr[Expt^{\texttt{EU-CMA}}_{\texttt{SGN}_2}(\mathcal{A}) = 1]$$

where $Pr[Expt^{\texttt{H-EU-CMA}}_{\texttt{HYSGN}}(\mathcal{A}) = 1] = Pr[Expt^{\texttt{EU-CMA}}_{\texttt{SGN}_1}(\mathcal{A}) = 1] \cdot Pr[Expt^{\texttt{EU-CMA}}_{\texttt{SGN}_2}(\mathcal{A}) = 1]$

The `EU-CMA` advantage of `SGN` is defined as

$$Adv^{\texttt{H-EU-CMA}}_{\texttt{HYSGN}}(t, q_s, q'_s) = \max_{\mathcal{A}}\{Adv^{\texttt{EU-CMA}}_{\texttt{HYSGN}}(\mathcal{A})\} = \min\{Adv^{\texttt{EU-CMA}}_{\texttt{SGN}_1}(t, q_s, q'_s)\ ,\ Adv^{\texttt{EU-CMA}}_{\texttt{SGN}_2}(t, q_s, q'_s)\},$$

Assumption 1: The commitment construct oracle (CCO) has a secure enclave as described in the system model. A post-quantum CCO is easily achieved by using PQ signature primitives within the encalve. The security of hardware-assisted signature primitives relies on trust in Intel SGX's manufacturing process and its robustness. The system could be also instantiated using other isolated execution environments (e.g., Sanctum [19]). It is crucial to recognize the limitations of relying on trusted execution environments. For example, Intel SGX encountered various side-channel attacks (e.g., [51]). Generic techniques for protection against enclave side-channel attacks are also under study in various works (e.g., [33]), therefore they are complementary to ours.

4 PROPOSED SCHEMES

Our goal is to create a new series of cryptographic tools for data authentication and integrity featuring resilient digital signatures with advanced security and performance attributes (e.g., FS, PQ security, aggregation) tailored for DT applications. However, designing a signature scheme that aligns with the stringent requirements of DT components poses a significant challenge [26]. Consider a human DT constantly receives data from various sources (e.g., wearable sensors, implantables) to replicate the state of the real twin. Yet, DT-IoT components face resource constraints (e.g., battery, processing) demanding efficient signature scheme. Additionally, DTs need a PQ signature with FS to guarantee long-term assurance and breach resiliency against malware compromises. It is also highly desirable to provide aggregation to reduce communication overhead. In the following, we propose three novel signature schemes that achieve all these seemingly conflicting performance and security goals.

4.1 Post-Quantum Hardware-Assisted Efficient Signatures (PQ-HASES)

Our initial contribution is an FS and PQ signature scheme with hardware support called PQ−HASES. As discussed in Section 1, the state-of-the-art PQ schemes are still very expensive to be deployed on low-end devices. To our knowledge, there is no open-source implementation of NIST PQC standards on highly resource-limited devices (e.g., 8-bit MCUs). It is even higher if FS is considered [58].

We created PQ−HASES to address these limitations. It primarily uses a hash-based one-time signa-ture scheme (HORS) [46], which is also the basis for NIST's recommendation XMSS^{MT}[18] and standard SPHINCS+ [8]. PQ−HASES achieves FS by updating the private key through a hash chain. The main bottleneck for hash-based schemes stems from an expensive computation and transmission of public keys. Thus, we introduce CCO as public-key issuer which enables a non-interactive signer that only broadcasts authenticated data. The algorithmic intuition of PQ−HASES is outlined below.

We give the detailed algorithmic description of PQ−HASES in Fig. 1.

In PQ−HASES.Kg, for a given set of users $\vec{ID}$, it first generates the master key msk and the CCO-related keys for key certification purposes (Step 1). It also accepts as input (J_1, J_2), the number of precomputed and interleaved keys, respectively. J represents the number of signatures to be generated (Step 2). According to J_1, it generates the precomputed keys $\vec{sk}_p^{\,n}$ from sk_1^n (Step 5-7). Each private seed sk_1^n is sent to its corresponding signer ID_n (Step 8), while $(msk, \vec{sk}_p)$ are placed on the secure enclave of CCO (Step 1).

PQ−HASES.ComConstr harnesses CCO to offer a trustworthy and flexible public commitment supply and identity management service for verifiers, without requiring interaction with signers. The verifier requests the one-time commitment C_j of ID for the state j. CCO first identify the corresponding pre-computed key (Step 1) and then derive the j^{th} secret key (Step 2). Finally, it generates the commitment C_j and returns it to the verifier (Step 3-5). CCO can derive any one-time commitment C_j of any $ID \in \vec{ID}$ on demand, making PQ−HASES fully scalable for millions of users with an adjustable $O(J_1)$ cryptographic data storage. Moreover, the verifier can obtain any public key(s) $1 \leq j \leq J$ from CCO in batches before receiving signatures (PQ−HASES.ComConstr is independent from signer). This permits verifiers to immediately verify signatures. Also, CCO can either present on the verifier

$(msk, \vec{sk}_1, \vec{sk}_p, I) \leftarrow$ PQ-HASES.Kg$(1^\kappa, \vec{ID} = \{ID_n\}_{n=1}^N, J_1, J_2)$:

1: Generate master key $msk \xleftarrow{\$} \{0,1\}^\kappa$ and CCO-related keys $(csk, CPK) \leftarrow$ SGN.Kg(1^κ). Securely store (msk, csk) at CCO.
2: Set $J \leftarrow J_1 \cdot J_2$ and $I \leftarrow (l, k, t, J_1, J_2, J)$
3: **for** $n = 1, \ldots, N$ **do**
4: $\quad sk_1^n \leftarrow \mathtt{PRF}(msk, ID_n)$
5: $\quad$ **for** $j_1 = 2, \ldots, J_1 - 1$ **do**
6: $\quad\quad sk^n_{j_1 \cdot J_2+1} \leftarrow H^{(J_2)}(sk^n_{(j_1-1)\cdot J_2+1})$
7: $\quad \vec{sk}_p^n \leftarrow \{sk^n_{j_1 \cdot J_2+1}\}_{j_1=1}^{J_1-1}$
8: $\vec{sk}_1 \leftarrow \{sk_1^n\}_{n=1}^N$, where each sk_1^n is provisioned to ID_n, as initial private key.
9: $\vec{sk}_p \leftarrow \{\vec{sk}_p^n\}_{n=1}^N$, are provisioned to CCO, as precomputed private keys.
10: **return** $(msk, \vec{sk}_1, \vec{sk}_p, I)$

$(C_j, \sigma_{C_j}) \leftarrow$ PQ-HASES.ComConstr$(msk, csk, \vec{sk}_p, ID, j)$: Require $ID \in \vec{ID}$ and $j \le J$

1: $j_1 \leftarrow \lfloor \frac{j-1}{J_2} \rfloor$ and $j_2 \leftarrow (j-1) \mod J_2$
2: Load $sk_{j_1 \cdot J_2+1}$ from $\vec{sk}_p$ and derive $sk_j \leftarrow H^{(j_2)}(sk_{j_1 \cdot J_2+1})$
3: **for** $\ell = 1, \ldots, t$ **do**
4: $\quad v_\ell \leftarrow H(s_\ell)$, where $s_\ell \leftarrow \mathtt{PRF}(sk_j, \ell)$
5: $C_j \leftarrow (v_1, v_2, \ldots, v_t)$
6: $\sigma_{C_j} \leftarrow$ SGN.Sig$(csk, v_1 \| \ldots \| v_t)$
7: **return** (C_j, σ_{C_j})

$\sigma_j \leftarrow$ PQ-HASES.Sig(sk_j, M_j): Init $(j = 1)$, require $j \le J$

1: $h \leftarrow H(M_j)$ and split h into k substrings $\{h_\ell\}_{\ell=1}^k$ such that $|h_\ell| = \log_2 t$, where each h_ℓ is interpreted as an integer x_ℓ
2: **for** $\ell = 1, \ldots, k$ **do**
3: $\quad s_\ell \leftarrow \mathtt{PRF}(sk_j, x_\ell)$
4: Update $sk_{j+1} \leftarrow H(sk_j)$, delete sk_j, and increment $j \leftarrow j+1$
5: **return** $\sigma_j \leftarrow (s_1, s_2, \ldots, s_k, j, ID)$

$b \leftarrow$ PQ-HASES.Ver$(\langle CPK, C_j, \sigma_{C_j} \rangle, M_j, \sigma_j)$: Steps 1-2 can run in offline mode.

1: $(C_j, \sigma_{C_j}) \leftarrow$ PQ-HASES.ComConstr$(msk, csk, \vec{sk}_p, ID, j)$
2: **if** SGN.Ver$(CPK, v_1 \| \ldots \| v_t, \sigma_{C_j}) = 1$ **then continue else return** 0
3: Execute Step (1) in PQ-HASES.Sig
4: **if** $H(s_\ell) = v_{x_\ell}, \forall \ell \in [1,k]$ and $1 \le j \le J$ **then return** $b = 1$ **else return** $b = 0$

Fig. 1. The proposed PQ hardware-assisted digital signature with forward-security (PQ-HASES)

machine (e.g., laptop), or a nearby edge-cloud server and therefore can effectively deliver public commitments on demand. In order to authenticate the commitments, we used a generic digital signature and generate a signature on the public commitment (Step 6).

The signing algorithm PQ-HASES.Sig relies on HORS signature but with a FS pseudo-random number generation. Given sk_j, the signer computes subset resilient indexes and derives the HORS one-time signature components (steps 1-4 in Fig. 1). The current private key is updated and the previous key is deleted. HASES.Sig offer near-optimal signing efficiency by running a single HORS call and a single hash for the update. The signer do not use or interact with secure hardware.

The algorithm PQ-HASES.Ver also depends on HORS, leveraging PQ-HASES.ComConstr to acquire certified public commitments which are verified via the CCO public key (Step 1-2). Upon having a valid commitment which represent the public key, it continues by performing the HORS signature verification algorithm (Step 3-4). It is important to note that PQ-HASES distinct itself from symmetric-key methods (e.g., MACs alone or use of secure hardware to compute/verify MACs) by providing public verifiability and non-repudiation. The verifier can validate the signatures with an offline interaction with CCO, which only supplies commitments and does not perform the verification.

Optimizations: Our design allows verifiers to request the one-time public keys before the signature verification, which avoids the network and CCO computation delays. PQ-HASES offers an adjustable storage-computation trade-off by controlling the number of pre-computed keys stored at CCO. We observed that CCO could benefit from an optimizer that manage the data storage overhead for each signer. For instance, this optimizer could be implemented with a Reinforcement Learning algorithm that learns from previous verifier requests [36] in order to reduce the computation delay.

4.2 Lightweight Aggregate Hardware-Assisted Efficient Signatures (LA-HASES)

In DT-enabled settings, extensive network traffic arises from end-to-end interactions between physical objects (e.g., humans, sensors) and their digital replicas (e.g., cloud server). Therefore, aggregation becomes pivotal in curbing

network delays, enhancing service quality, conserving transmission energy in wireless setups, and consequently prolonging battery life. Moreover, it alleviates local storage on signers, thus creating more room for the main core-specific DT applications.

Algorithmic Novelty and Differences with Previous Works: To the best of our knowledge, no aggregate PQ signature exists that is computation and bandwidth efficient. For example, a recent lattice-based AS scheme [12] offer logarithmic compression w.r.t number of signatures. However, it is limited to either one-time non-interactive or multiple-time interactive setting. Another approach converts conventional aggregate signatures into the lattice domain via Fiat-Shamir with Aborts [14]. Yet, this incur higher computational overhead and minimal compression gains.

Among conventional-secure alternatives, recent IoT authentication protocols (e.g., [34, 52]) predominantly rely on the primitive pairing-based AS scheme, BLS [13]. As outlined in Section 1.1, BLS impose a burdensome signing overhead on resource-constrained devices, primarily due to costly map-to-point and modular exponentiation operations it entails. Furthermore, these works typically lack performance assessments on low-end MCUs, leaving their practicality unjustified.

EC-based AS schemes (e.g., [39, 42, 59]) offer a balance between compact signature size and efficient signing when compared to above alternatives. However, it comes at the cost of linear communication and storage overhead for one-time commitments. In fact, FI-BAF [59] tranforms Schnorr [20] into an an aggregate digital signature by replacing $H(M\|R)$ with $H(M\|x)$, where x is a one-time random key. This allows detaching the commitment computation from signature generation and performing it during key generation in offline mode. Note that the main bottleneck in signature generation of Schnorr-based schemes lies in calculating the commitment R which entails expensive modular exponentiation per signing. This is why FI-BAF offer a significantly more cost-effective signature generation compared to Schnorr. Nevertheless, verifiers are required to store linear one-time commitments w.r.t number of messages. This makes it not scalable for our specific DT applications.

`LA-HASES` aims to optimize FI-BAF [59], which represents the improved version of the primitive Schnorr [20], upon which recent lightweight authentication protocols (e.g., [52]) depends. Aligned with `PQ-HASES`, we create a Lightweight Aggregate HArdware-Assisted Efficient Signature (`LA-HASES`) that uses `CCO` to transmit costly EC commitments to verifiers, enabling a non-interactive efficient aggregate signing. Therefore, `LA-HASES` achieve near-optimal signing with few modular arithmetic operations and hash calls while also leaving verifers with low and flexible storage overhead. That is, verifiers can request commitments in batches in offline mode or on-demand. Unlike FI-BAF, `CCO` supplies verifiers with constant-size aggregate commitments per batch of input messages which significantly lower the communication and storage overhead on verifier side. Moreover, `CCO` can live on verifiers, thereby enabling zero communication overhead.

We give a detailed algorithmic description of `LA-HASES` in Fig. 2.

In `LA-HASES.Kg`, it generates system parameters, which include the EC-based settings, the maximum number of generated signatures J, and the batch size L (Step 1). For a given set of users $\vec{ID}$, it generates the master key msk and `CCO` related keys (Step 2) (provisioned to `CCO`), derives the users' private keys $\vec{sk}$ and provision each private key sk_n to its corresponding user ID_n (Step 3-5).

`LA-HASES.ComConstr` offers a secure commitment supply service for the verifiers. In offline or on-demand mode, the verifier can request the public commitments of any user ID_n and for any given state j. `CCO` identifies the user's private key and computes r_j of the requested epoch j (Step 1). Then, it constructs the aggregate public commitment $R_j^{1,L}$ (Step 2), and returns it as output (Step 3).

`LA-HASES.ASig` computes a fixed-length signature for a set of L messages. First, it computes the seed x_j and commitment r_j for epoch j (Step 1). The aggregate signature $s_j^{1,L}$ is computed without expensive EC scalar multiplications, relying on few `PRF` and hash calls, and arithmetic operations per item (Step 3-6). Finally, the signer state is updated and the signature is returned (Steps 7-8).

$(msk, \vec{sk}, \vec{PK}, I) \leftarrow$ LA-HASES.Kg$(1^\kappa, \vec{ID} = \{ID_n\}_{n=1}^N, J, L)$:

1: Generate large primes q and $p \geq q$ such that $q|(p-1)$. Select a generator α of the subgroup G of order q in $\mathbb{Z}_q^*$. Set $I \leftarrow (p, q, \alpha, J, L, St : (ID_n\ ,\ j = 1)),\ \forall n \in [1, N]$
2: Generate master key $msk \xleftarrow{\$} \{0,1\}^\kappa$ and CCO-related keys $(csk, CPK) \leftarrow$ SGN.Kg(1^κ). Securely store (msk, csk) at CCO.
3: **for** $n = 1, \ldots, N$ **do**
4: $\quad y_n \leftarrow$ PRF$(msk, ID_n) \mod q$, and $Y_n \leftarrow \alpha^{y_n} \mod p$
5: $\vec{sk} \leftarrow \{csk, y_1, \ldots, y_N\}$, where y_n is provisioned to ID_n.
6: $\vec{PK} \leftarrow \{CPK, Y_1, \ldots, Y_N\}$
7: **return** $(msk, \vec{sk}, \vec{PK}, I)$

$\delta_{1,u} \leftarrow$ LA-HASES.Agg$(\{\delta_\ell \in \sigma\}_{\ell=1}^u)$:

1: **if** $\delta \in \mathbb{Z}_q^*$ **then** $\delta_{1,u} \leftarrow \sum_{\ell=1}^u \delta_\ell \mod q$
2: **return** $\delta_{1,u}$

$(C_j^{1,L}, \sigma_{C_j}) \leftarrow$ LA-HASES.ComConstr$(msk, csk,\ ID, j)$: Require $ID \in \vec{ID}$ and $j \leq J$

1: $y \leftarrow$ PRF$(msk, ID) \mod q$, and $r_j \leftarrow$ PRF$(y, j\|2) \mod q$
2: $R_j^{1,L} \leftarrow \alpha^{\sum_{\ell=1}^L r_j^\ell \mod q} \mod p$, where $r_j^\ell \leftarrow$ PRF$(r_j, \ell) \mod q, \forall \ell \in [1, L]$
3: $\sigma_{C_j} \leftarrow$ SGN.Sig$(csk, C_j^{1,L})$, where $C_j^{1,L} \leftarrow R_j^{1,L}$
4: **return** $(C_j^{1,L}, \sigma_{C_j})$

$\sigma_j^{1,L} \leftarrow$ LA-HASES.ASig$(y, \vec{M}_j)$: require $j \leq J$ and $\vec{M}_j = (m_j^1, \ldots, m_j^L)$

1: $x_j \leftarrow$ PRF$(y, j||1)$, and $r_j \leftarrow$ PRF$(y, j\|2)$
2: $s_j^{1,0} \leftarrow 0$
3: **for** $\ell = 1, \ldots, L$ **do**
4: $\quad x_j^\ell \leftarrow$ PRF(x_j, ℓ), and $r_j^\ell \leftarrow$ PRF$(r_j, \ell) \mod q$
5: $\quad s_j^\ell \leftarrow r_j^\ell - e_j^\ell \cdot y \mod q$, where $e_j^\ell \leftarrow H(m_j^\ell \| x_j^\ell)$
6: $\quad s_j^\ell \leftarrow$ LA-HASES.Agg$(s_j^{1,\ell-1}, s_j^\ell)$
7: Update $St : (ID\ ,\ j \leftarrow j+1)$
8: **return** $\sigma_j^{1,L} \leftarrow \langle s_j^{1,L}, x_j, St \rangle$

$b \leftarrow$ LA-HASES.AVer$(\langle CPK, PK, C_j^{1,L} \rangle, \vec{M}_j, \sigma_j^{1,L})$: require $\vec{M}_j = \{m_j^\ell\}_{\ell=1}^L$. Steps 1 can be run offline.

1: $(C_j^{1,L}, \sigma_{C_j}) \leftarrow$ PQ-HASES.ComConstr(msk, csk, ID, j), where $C_j^{1,L} = R_j^{1,L}$ (offline mode or on-demand)
2: **if** SGN.Ver$(CPK, C_j^{1,L}, \sigma_{C_j}) = 1$ **then continue else return** 0
3: Set $e_j^{1,0} \leftarrow 0$
4: **for** $\ell = 1, \ldots, L$ **do**
5: $\quad x_j^\ell \leftarrow$ PRF(x_i^ℓ, ℓ)
6: $\quad e_j^{1,\ell} \leftarrow e_j^{1,\ell-1} + e_j^\ell \mod q$, where $e_j^\ell \leftarrow H(m_j^\ell \| x_j^\ell)$
7: **if** $R_j^{1,L} = Y_j^{e_j^{1,L}} \cdot \alpha^{s_j^{1,L}}$ **then return** $b = 1$ **else return** $b = 0$

Fig. 2. The proposed lightweight aggregate-based hardware-assisted digital signature (LA−HASES)

LA−HASES.AVer can receive the certified public commitments from CCO either in offline or on-demand mode (Step 1). It computes the one-time ephemeral keys $\{e_j^\ell\}_{\ell=1}^L$ to recover the aggregate key $e_j^{1,L}$ (Step 3-6). Finally, it checks the verification equation and outputs a validation bit (Step 7).

4.3 HYbrid Hardware-ASsisted Efficient Signatures (HY-HASES)

The migration from conventional-secure standards to PQC is expected to be a significant effort, posing challenges due to the heterogeneous nature of current software/hardware platforms. Given the substantial expenses and risks associated with PQC (e.g., algorithmic breaks), NIST recommends hybrid architectures as a transitional strategy, that fuses conventional and PQ schemes to leverage their distinct attributes [5]. Hybrid signatures promote cryptographic agility which is crucial to resist single points of failures [40]. It offers a better resiliency to emerging hardware technology and algorithmic breakthroughs like recently broken NIST PQC candidates. Moreover, the hybrid solution should ideally be standard-compliant for ease of transition. Our main idea is to leverage conventional-secure aggregate LA−HASES to enhance signing efficiency, complemented with PQ−HASES for per-batch FS and PQ securities. This results in our proposed HY−HASES digital signature. It is a combination of LA−HASES and PQ−HASES, enriched by a strong novel nesting technique.

Algorithmic Novelty and Differences with Previous Works: As discussed in Section 1.1, recent hybrid authentication schemes fail to address practicality issues on resource-constrained networks. For example, in a benchmark study [43], a variety of hybrid digital signatures were tested, using both standard-compliant conventional (e.g., ECDSA) and PQ (e.g., Dilithium) digital signatures. However, our findings demonstrate that deploying standard digital signatures on low-end MCUs is impractical due to their high signature generation costs. Furthermore, standard digital signatures lack essential security features, specifically aggregation and forward security, which are crucial for optimizing network throughput and safeguarding against physical attacks.

Hybrid digital signatures involve merging various digital signatures through a nesting algorithm to achieve security of all underlying signature schemes. However, existing combination methods (e.g., [9]) fail to handle merging digital signatures with varying security properties. To our knowledge, no prior work has tackled the integration of an AS scheme with an FS and PQ digital signature. Nonetheless, we recognize the importance of simultaneously achieving these security features while maintaining efficient signature generation, especially for low-end IoT devices. In this context, we present a novel nesting strategy that uniquely blend LA−HASES and PQ−HASES signature schemes to construct the hybrid HY−HASES, achieving all desirable properties.

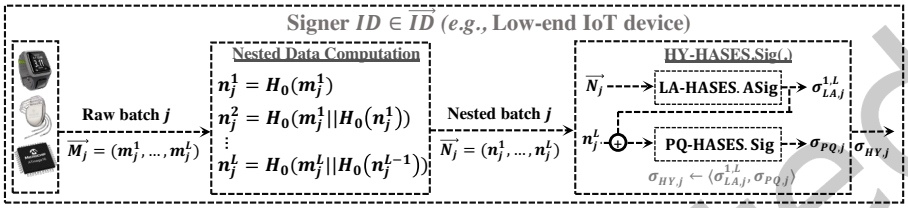


Fig. 3. High-level description of HY−HASES signature generation algorithm

We give a detailed algorithmic overview of HY−HASES in Fig. 4 and high-level depiction in Fig. 3. We adopt the aggregate LA−HASES for intra-batch input messages, in order to minimize both the cryptographic payload and the energy use during continuous data offload. To achieve the PQ and FS features, we utilize PQ−HASES for inter-batches. HY−HASES key generation involve both LA−HASES.Kg and PQ−HASES.Kg algorithms. Given set of users $\vec{ID}$, HY−HASES.Kg initially generates master key and system parameters for PQ−HASES and LA−HASES (Step 1-2). The master secret key of HY−HASES is composed of msk_{LA} and msk_{PQ}. The private key vector $\vec{sk}$ and the system parameters I are also compromised of LA and PQ components.

Combiners enable the construction of hybrid signatures. For example, strong nesting (see Def. 2.4) involves one scheme generating a signature on an input message, which is then passed alongside the message as input to the second scheme. Yet, they are not applicable to merging the underlying LA−HASES and PQ−HASES schemes. Indeed, it is inefficient to upload a data batch (i.e., $O(L)$) to the FS and PQ scheme (PQ−HASES) as input due to the $O(L)$ linear signature overhead.

To counter this, we construct a nested vector $\vec{N}_j$ sequentially from the raw input vector $\vec{M}_j = \{m_j^\ell\}_{\ell=1}^L$. This envolves concatenating the current message m_j^ℓ with the hash of the previous nested element $H_0(n_j^{\ell-1})$ and hashing the resulting string (see Fig. 3 and HY−HASES.Sig in Fig. 4). The elements of $\vec{N}_j$ are: $n_j^1 = H_0(m_j^1)$ and $n_j^\ell = H_0(m_j^\ell \| H_0(n_j^{\ell-1}))$, $\forall \ell = 2, \ldots, L$. The final element n_j^L represents a *holistic digest* of the data batch $\vec{M}_j$. In HY−HASES.Sig, we generate the aggregate signature $\sigma_{LA,j}^{1,L}$ (Step 1). Next we compute the PQ signature by inputting the concatenation of the aggregate signature and the last element of the nested data (i.e., $s_j^{1,L} \| n_{j,L}$) to the PQ−HASES.Sig scheme. The resulting hybrid signature $\sigma_{HY,j}^{1,L}$ consists of the concatenation of both signatures $\sigma_{LA,j}^{1,L}$ and $\sigma_{PQ,j}$. For the signature verification via HY−HASES.Ver, it receives one-time public commitments for both schemes (Step 1). The hybrid signature is verified only if both of the LA−HASES and PQ−HASES verification algorithms are valid (Step 2).

5 SECURITY ANALYSIS

We formally prove that PQ−HASES, LA−HASES, and HY−HASES are F−EU−CMA, A−EU−CMA, and H−EU−CMA secure (in the random oracle model [32]) in Theorem 5.1, 5.2, and 5.3, respectively. We ignore the terms that are negligible in terms of κ. The full proofs are given in Appendix 7. Our claimed security proofs

$(msk_{HY}, \vec{sk}, I) \leftarrow$ HY-HASES.Kg$(1^\kappa, \vec{ID} = \{ID_n\}_{n=1}^N, J, L)$:

1: $(msk_{LA}, \vec{sk}_{LA}, \vec{PK}_{LA}, I_{LA}) \leftarrow$ LA-HASES.Kg$(1^\kappa, \vec{ID} = \{ID_n\}_{n=1}^N, J, L)$
2: $(msk_{PQ}, \vec{sk}_{PQ}, \vec{sk}_{pPQ}, I_{PQ}) \leftarrow$ PQ-HASES.Kg$(1^\kappa, \vec{ID} = \{ID_n\}_{n=1}^N, J_1, J_2)$, where $J = J_1 \cdot J_2$
3: $msk_{HY} \leftarrow (msk_{LA}, msk_{PQ}, \vec{sk}_{pPQ}, \vec{sk}_{pLA})$, $\vec{sk}_{HY} \leftarrow (\vec{sk}_{LA}, \vec{sk}_{PQ})$, and $I \leftarrow (I_{LA}, I_{PQ})$
4: **return** $(msk_{HY}, \vec{sk}_{HY}, I)$

$(C^{1,L}_{LA,j}, \sigma_{C_{LA,j}}, C_{PQ,j}, \sigma_{C_{PQ,j}}) \leftarrow$ HY-HASES.ComConstr$(msk_{HY}, csk_{HY}, ID, j)$: Require $ID \in \vec{ID}$ and $j \leq J$

1: $(C^{1,L}_{LA,j}, \sigma_{LA,j}) \leftarrow$ LA-HASES.ComConstr$(msk_{LA}, csk_{LA}, ID, j)$
2: $(C_{PQ,j}, \sigma_{C_{PQ,j}}) \leftarrow$ PQ-HASES.ComConstr$(msk_{PQ}, \vec{sk}_{pPQ}, ID, j)$
3: **return** $(\langle C^{1,L}_{LA,j}, \sigma_{C_{LA,j}}\rangle, \langle C_{PQ,j}, \sigma_{C_{PQ,j}}\rangle)$

$\sigma_{HY,j} \leftarrow$ HY-HASES.Sig$(\vec{sk}, \vec{N}_j)$: require $j \leq J$ and $\vec{N}_j = (n^1_j = H(m^1_j), \ldots, n^L_j = H(m^L_j \| H(n^{L-1}_j)))$

1: $\sigma^{1,L}_{LA,j} \leftarrow$ LA-HASES.ASig$(sk_{LA,j}, \vec{N}_j)$, where $\sigma^{1,L}_{LA;j} = \langle s^{1,L}_j, x_j, ID, j\rangle$
2: $\sigma_{PQ,j} \leftarrow$ PQ-HASES.Sig$(sk_{PQ,j}, s^{1,L}_j \| n_L)$
3: **return** $\sigma_{HY,j} \leftarrow \langle \sigma^{1,L}_{LA,j}, \sigma_{PQ,j}\rangle$

$b \leftarrow$ HY-HASES.Ver$(\langle PK_{HY,j}, C_{HY,j}\rangle, \vec{N}_j, \sigma_j)$: Step 1 can be run offline.

1: $(\langle C^{1,L}_{LA,j}, \sigma_{C_{LA,j}}\rangle, \langle C_{PQ,j}, \sigma_{C_{PQ,j}}\rangle) \leftarrow$ HY-HASES.ComConstr$(msk_{HY}, csk_{HY}, ID, j)$
2: **if** SGN.Ver$(CPK_{PQ}, C_{PQ,j}, \sigma_{C_{PQ,j}}) = 1$ **and** SGN.Ver$(CPK_{LA}, C_{LA,j}, \sigma_{C_{LA,j}}) = 1$ **then continue else return** 0
3: **return** (LA-HASES.AVer$(\langle PK_{LA,j}, C^{1,L}_{LA,j}\rangle, \vec{M}_j, \sigma^{1,L}_{LA,j})) \wedge$ (PQ-HASES.Ver$(\langle PK_{PQ,j}, C_{PQ,j}\rangle, s^{1,L}_{LA,j} \| n_L, \sigma_{PQ,j})$)

Fig. 4. The proposed hybrid hardware-assisted digital signature (HY-HASES)

follows the seminal works (e.g., [32]). Hence, it inherits similar limitations related to unsufficient security model or unpractical assumptions [54].

THEOREM 5.1. *If a polynomial-time adversary $\mathcal{A}$ can break the F-EU-CMA secure PQ-HASES in time t and after q_s signature and commitment queries, and q'_s queries to RO(.), with a break-in query, then one can build polynomial-time algorithm $\mathcal{F}$ that breaks the EU-CMA secure HORS in time t' and q'_s queries under Assumption 1.*

$$Adv^{F-EU-CMA}_{PQ-HASES}(t, q_s, q'_s, 1) \quad \leq \quad J \cdot Adv^{EU-CMA}_{HORS}(t', q_s, q'_s)\ ,\ \textit{where}\ q'_s = q_s + 1\ \textit{and}\ O(t') = O(t) + k \cdot H_0$$

THEOREM 5.2. *If a polynomial-time adversary $\mathcal{A}$ can break the A-EU-CMA secure LA-HASES in time t and after q_s signature and public key queries, and q'_s queries to the random oracle RO(.), the one can build polynomial-time algorithm $\mathcal{F}$ that breaks the DLP problem in time t' and q'_s queries under Assumption 1.*

$$Adv^{A-EU-CMA}_{LA-HASES}(t, q_s, q'_s) \leq Adv^{DL}_{G,\alpha}(t')\ ,\ \textit{where}\ t' = O(t) + O(\kappa^3)$$

THEOREM 5.3. *If either LA-HASES or PQ-HASES is unforgeable in the classical (or quantum) random oracle model, then the hyrbid scheme HY-HASES is unforgeable in the classical (or quantum) oracle model, respectively, under Assumption 1.*

$$Adv^{H-EU-CMA}_{HY-HASES}(t, q_s, q'_s) = \min\{Adv^{F-EU-CMA}_{PQ-HASES}(t, q_s, q'_s, 1)\ ,\ Adv^{A-EU-CMA}_{LA-HASES}(t, q_s, q'_s)\}$$

6 PERFORMANCE ANALYSIS

This section examines the performance of HASES schemes and contrasts them with their counterparts.

6.1 Evaluation Metrics and Experimental Setup

Evaluation metrics: We compare HASES schemes and their counterparts based on: (i) signer's efficiency (i.e., key sizes, computation, energy use) (ii) advanced security aspects (e.g., FS, PQ security), (iii) compliance with standards and ease of transition, (iv) verifier's computational overhead.

Selection Rationale of Compared Counterparts: We selected counterparts to represent the performance of primary conventional-secure aggregate and PQ digital signature families. Specifically, we consider the following conventional aggregate signatures: *(i) Pairing-based:* BLS [13] is based on pairing maps. It is well-suited for multi-user scenarios, characterized by compact signature and public key sizes. *(ii) Factorization-based:* C-RSA [57] providing mainly a fast signature verification.

For PQ signatures, we consider: *(i) Lattice-based:* Dilithium [24] is a NIST PQC standard, featur-ing efficient signing with moderate key sizes compared to other PQC candidates. BLISS [27] is the sole PQ signature with an open-source implementation for low-end devices (i.e., 8-bit AVR MCU). *(ii) Hash-based:* SPHINCS+ [8] is the sole hash-based NIST PQC standard. Also, XMSS^{MT} [18] is a FS signature, serving as both an RFC standard and a NIST recommendation.

We assess the performance of HY−HASES by comparing it to a nested combination of the most relevant conventional aggregate and PQ signatures from our listed counterparts. We note that numerous orthogonal efforts targeted IoT authentication as discussed in Section 1.1. Many of these works use digital signatures as a fundamental component, thus potentially benefitting from our schemes. Therefore, our focus remains on comparing HASES to combinations of above digital signatures.

Parameter Selection: We set the security parameter as $\kappa = 128$ and SHA-256 as our cryptographic hash function H. We used Ascon as our PRF for key derivation calls. Ascon have been selected as the NIST standard for lightweight cryptography since it offer high efficiency and security measures. We used the Curve25519 [7] (as NIST's FIPS 186-5 standard, 256-bit public keys) for our EC-based LA−HASES scheme to offer standard compliance and a better computational efficiency. We set the number of signers as $N = 2^{20}$ and the maximum number of messages to be signed as $J = 2^{20}$ (as in XMSS [30]). The user identity list $\vec{ID} = \{ID_i\}_{i=1}^{N}$ are considered as MAC addresses. In PQ−HASES, we choose $I_{PQ} \leftarrow \{l = 256, t = 1024, k = 16\}$ for a security level $\kappa = 128$. Our choice prioritizes an optimal signer efficiency as it only requires a few (≈ 20) hash calls to perform one signature generation. The resource-limited signers are non-interactive and do not communicate public keys, and therefore the parameter t does not impact the signing performance. The composite modulo size in C-RSA is $|n| = 2048$. The generic digital signature SGN that is used in the commitment construction algorithm is instantiated with the NIST PQC standard Dilithium [24].

6.2 Performance Evaluation and Comparison

6.2.1 Performance on Commodity Hardware. We now outline the evaluation of signer and verifier sides using the commodity hardware.

Hardware and Software Configuration: HASES schemes were fully implemented on the signer and verifier sides using commodity hardware. We used a desktop with an Intel i9-9900K@3.6 GHz processor and 64 GB of RAM. Our approach is based on OpenSSL[2] and Intel SGX SSL[3] open-source libraries. The benchmarking schemes utilized a dataset[4] with samples from accelerometers, gyroscopes, and ECG samples collected from 29 volunteers, simulating a DT-enabled health monitoring application. This involves secure offloading of data to a cloud for disease detection. The average message size is 32 bytes, with minimal extra overhead for larger messages (due to hash compression).

[2]https://github.com/openssl/openssl

[3]https://github.com/intel/intel-sgx-ssl

[4]https://ieee-dataport.org/documents/mechanocardiograms-ecg-reference

Performance Analysis: Table 2 illustrates the overall performance of HASES schemes and their counterparts at the signer and verifier sides. We present the main takeaways as follows:

• *Signature Generation:* The conventional-secure aggregate LA−HASES scheme offers a speedup of 762× and 183× over the pairing-based BLS and factorization-based C-RSA schemes, respectively. This speedup is attributed to LA−HASES's distinct commitment separation via CCO. In contrast, BLS and C-RSA necessitate resource-intensive map-to-point and scalar multiplication operations, respectively, both scaling linearly with batch size. PQ−HASES also outperforms its counterparts with 1542× speedup over its only FS and PQ hash-based counterpart, XMSS^{MT} [18]. It is also 16.28× faster than the non-forward-secure NIST PQC standard, Dilithium-II [24]. PQ−HASES achieves this while executing only a few hash and PRF calls (≈ 20), ensuring robust security and fast signing. Additionally, it exhibits a 74× signing speedup over BLISS-I which lacks NIST PQC standardization and remains vulnerable to devastating side-channel and timing attacks [27]. We further investigate the signing capabilities of HASES schemes by examining the signature generation in relation to variations in the input message size. As illustrated in Fig. 5, PQ−HASES surpasses its most efficient counterpart, Dilithium [24], by several orders of magnitude. We conducted benchmark tests on HASES schemes using two different hash instantiations: the NIST standard SHA256 and the NIST lightweight PRF function Ascon. We observe that opting for Ascon as the PRF choice results in relatively higher speedup compared to LA−HASES. This is attributed to the frequent hash and PRF calls for the hash-based HASES. We conducted a comparison between LA−HASES and the standard Ed25519, demonstrating the performance advantages offered by LA−HASES.

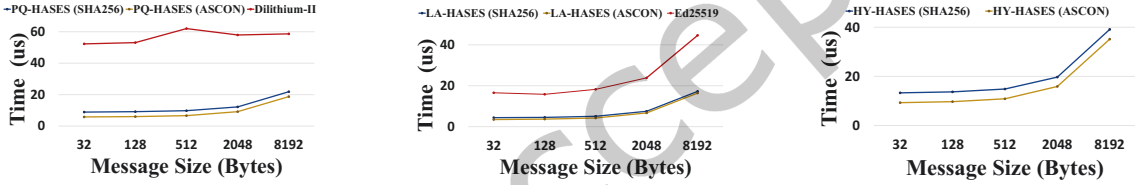


Fig. 5. Signature generation of HASES schemes and their counterparts on Commodity Hardware

• *Signature Size:* LA−HASES provides a compact signature, 4× smaller than RSA-2048 and equal in size to Ed25519. The signature size of PQ−HASES is 195× and 16× smaller than the forward-secure XMSS^{MT} and the NIST PQC standard Dilithium, respectively.

• *Signature Verification and Verifier Storage:* The delay of the CCO computation is parameterized by Δ in PQ−HASES whereas in the aggregate LA−HASES scheme it is constant. Verifiers can request one-time commitments in either *Offline* or *Online* mode. In offline mode, the CCO delay does not affect signature verification, resulting in significant gains compared to our counterparts. For example, if the commitment's retrieval is offline, PQ−HASES verification is 11× and 622× much faster than Dilithium-II (non-forward-secure) and XMSS^{MT} (forward-secure), respectively. The commitment construction delay is determined by the number of precomputed PQ−HASES private keys (i.e., J_1) stored at CCO. Fig. 6-(a) depicts the variation of this CCO storage on the ComConstr computation delay. For minimal storage of 320 MB in the secure enclave within CCO, the delay is the highest at $\Delta = 63.64$ msec, making the offline mode the recommended option. In contrast, if only 2^{10} precomputed keys are stored for each user in the network, the CCO storage increases to 32 GB, resulting in a significant improvement in the ComConstr delay with $\Delta = 7.13$ msec during online requests. We note that 32 GB represents only 6.25% of the protected memory in the second generation of SGX, which is widely integrated into new Intel Ice Lake processors [25].

6.2.2 Performance on 8-bit AVR Microcontroller. Table 3 compare the signer performance of HASES schemes with their counterparts on the 8-bit AVR microcontroller.

Table 2. Performance comparison of HASES variants and its counterparts on a commodity hardware

Scheme	Signing Time (μs)	Priv Key	Sig Size	Pub Key	Ver‡ Time (μs)	Verifier⋆ Storage (GB)	Long-Term Security	Backward Compatible	Ease of Transition	FSec	Agg Sig
Ed25519 [7]	31.85	0.03	0.06	0.03	78.99	0.09	○	✓	×	×	×
RSA-2048 [57]	413.36	0.5	0.25	0.5	21.84	0.75	○	✓	×	×	✓
BLS [13]	1,722.62	0.06	0.03	0.06	3,568.17	0.09	○	×	×	×	✓
LA-HASES	**2.26**	**0.03**	**0.05**	**0.03**	**431.4**	**32 B**	○	×	×	×	✓
BLISS-I [27]	252.35	2.00	5.60	7.00	25.02	12.6	●	×	×	×	×
Dilithium-II [24]	55.51	2.53	2.36	1.28	18.65	3.64	●	×	×	×	×
SPHINCS+ [8]	4,712.23	0.1	35.66	0.05	331.4	35.71	●	×	×	×	×
XMSS^{MT} [18]	5,213.93	5.86	4.85	0.06	1,057.63	4.91	●	×	×	✓	×
PQ-HASES	**3.41**	**0.03**	**0.5**	**32**	**1.7 + Δ**	**32 B + D**	●	✓	×	✓	×
Hybrid Aggregate Signature Constructions (numbers are for L items)											
RSA ($L\times$) + Dilithium	424,882.39	6.53	2.45	4.48	476.71	6.93	◑	×	✓	×	✓
RSA ($L\times$) + XMSS^{MT}	430,040.81	7.11	2.64	4.11	5635.13	6.75	◑	×	✓	✓	✓
BLS ($L\times$) + Dilithium	120,650.65	2.78	4.42	7.73	122,155.19	12.15	◑	×	✓	×	✓
BLS ($L\times$) + XMSS^{MT}	120,650.65	3.26	4.61	7.36	127,313.61	11.97	◑	×	✓	✓	✓
HY-HASES ($L\times$)	**1100.04**	**1.88**	**0.55**	**374.02**	**644.83 + Δ**	**64 B + D**	◑	✓	✓	✓	✓

The private/public key and signature sizes are in KB. We benchmarked the XMSSMT-SHA20/2_256 variant which allows for 2^{20} messages to be signed. For SPHINCS+ parameters, $n = 16, h = 66, d = 22, b = 6, k = 33, w = 16$ and $\kappa = 128$. The batch size of input messages is $L = 1024$. We set $N = 2^{20}$ signers and $J = 2^{10}$ as signing capability per user. $L\times$ refers to the batch size, fed to the AS schemes. ○ and ● denotes conventional and post-quantum security guarantees, respectively. ◑ denotes PQ-conventional hybrid security. BLISS and Dilithium incurs sampling operations which may be costly for low-end devices and prone to side-channel attacks [27].

⋆ Our system model assumes that the edge cloud performs signature verification to further analyze the multimedia data. In such case, CCO lives on verifiers. We consider that the verifier storage for HASES schemes consists of CCO private keys. The verifier storage for our counterparts is the cryptographic keys (i.e., public key and its certificate) for all users in the network (i.e., $N = 2^{20}$). If CCO does not live on verifier, we assume the verifier stores users' public keys or obtains them along one-time commitments from CCO, online or offline.

‡ Δ denotes the CCO delay of commitment construction in PQ-HASES. Δ depends on the storage of precomputed PQ-HASES private keys by CCO, which is defined by D. Fig. 6-(a) depicts the impact of CCO storage on the ComConstr computation overhead.

Hardware and software configuration: We implemented HASES schemes on 8-bit MCU at the signer end using an Arduino Mega2560 board. This board features an 8-bit ATMega2560 MCU with 256KB flash memory, 8KB SRAM and 4KB EEPROM operating at 16MHz clock frequency. We used the μNaCl open-source software library [5]. To demonstrate the cryptographic overhead in the low-end platforms, we use pressure sensor as our data source (see Fig. 6) to simulate the sensory capabilities of twins equipped with sensors and actuators that replicate various bodily functions. We compared the energy usage between one signature generation and a sensor sampling under different frequencies. Our goal is evaluating the trade-off between application and cryptographic overhead.

Performance Analysis: Table 3 showcases the performance comparison of HASES schemes and their counterparts on the 8-bit MCU. We present the main takeaways as follows:

For a batch of 1024 messages, LA-HASES.ASig offer an aggregate signature generation that is 9× more efficient and only 6.5× slower than an individual generation of an RSA and Ed25519 signature, respectively, computed on a single input. Unlike RSA, which introduces latency for batch processing, LA-HASES is better suited for resource-constrained devices and massive data offload due its use of Curve25519, ensuring standard compliance and backward compatible. Moreover, LA-HASES offer a stateful digital signature that only trigger the pseudo-random number generator once during the key generation. Thus, it avoids the vulnerabilities that stems

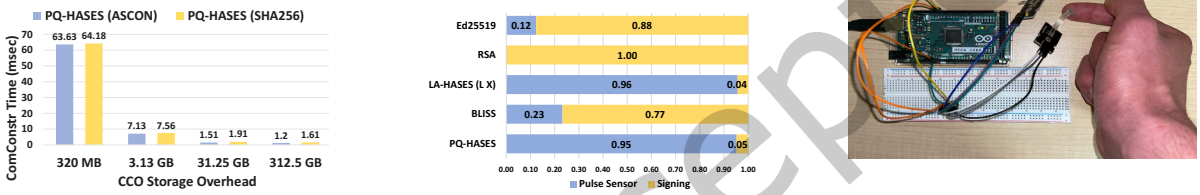
[5] http://munacl.cryptojedi.org/index.shtml

from the weak pseudo-random generators on resource-constrained devices. Furthermore, LA−HASES generate distinct commitments for each state, thereby ensuring tighter security reduction.

Table 3. Performance analysis of HASES schemes and counterparts on AVR ATmega2560 MCU

Scheme	Signing Time (sec)	Private Key (KB)	Signature Size (KB)	PQ Promise	Standard Compliant	Backward Compatible	Forward Security	Agg Capability
Ed25519 [7]	1.42	0.03	0.06	×	✓	✓	×	×
RSA-2048 [57]	83.26	0.5	0.25	×	✓	✓	×	✓
LA-HASES ($L\times$)	**9.24**	**0.03**	**0.05**	×	✓	✓	×	✓
⋆BLISS-I [27]	0.66	2.00	5.6	✓	×	×	×	×
PQ-HASES	**0.01**	**0.03**	**0.5**	✓	✓	✓	✓	×
HY-HASES ($L\times$)	**9.25**	**0.06**	**0.55**	✓	✓	✓	✓	✓

EPID [11] and SCB [41] are excluded as they are not inline with our system model (i.e., signers are low-end devices lacking secure enclaves).
⋆ BLISS suffer from rejection sampling which results in devastating side-channel attacks [27]. L denotes the batch size ($L = 1024$).
‡ FI-BAF incur a large storage overhead at the verifier side that is linear w.r.t the number of items to be signed.



(a) CCO Storage and Delay (b) Energy usage (c) Implementation Setup

Fig. 6. CCO Storage Impact on ComConstr Delay and Energy Usage of HASES schemes

We identify only one PQ signature scheme, BLISS [27], with benchmarking on 8-bit MCUs. Our PQ−HASES is 66× faster than BLISS-I, which is vulnerable from side-channel attacks due to rejection sampling [27]. PQ−HASES is 142× faster than standard Ed25519, which is neither PQ-secure nor forward-secure. PQ−HASES is also standard compliant and backward-compatible.

Among conventional-secure and PQ hybrids, HY−HASES stands as the sole feasible choice for DT-enabled platforms, especially on low-end devices, ensuring compact key sizes for continuous authentication. HY−HASES is backward compatible and adheres to standards by leveraging SHA-256 and EC-based operations on Curve25519. HY−HASES offers a highly efficient aggregate signature, complemented with a PQ and FS umbrella signature using minimal hash calls. Therefore, our experiments affirms HY−HASES as the ideal hybrid signature scheme for DT applications.

7 CONCLUSION

In this paper, we introduce Hardware-ASsisted Efficient Signatures (HASES) to address the critical requirements of lightweight digital signatures with exotic features in Digital Twins (DTs) applications. The HASES schemes, including PQ−HASES, LA−HASES, and HY−HASES, leverage a hardware-assisted cryptographic commitment construct oracle (CCO) to enable efficient verifiability without direct interaction with signers. These schemes offer features such as quantum-safe forward security, aggregate EC signatures, and hybrid conventional-secure and PQ combinations. By specifically targeting the resource limitations of low-end IoT devices commonly found in DT systems, HASES ensure long-term security and breach resiliency. Experimental results demonstrate the superior signer efficiency of PQ−HASES and LA−HASES compared to their PQ and conventional-secure counterparts,

respectively. The contributions of this work address the pressing need for lightweight digital signatures in DTs, providing both advanced security guarantees and efficiency. Through formal security proofs and comprehensive implementations, HASES represent a significant advancement in ensuring the trustworthiness of sensitive information in DT applications. By open-sourcing the HASES schemes, we invite further exploration, testing, and adaptation, facilitating progress in trustworthy DTs and enabling an effective transition into the PQ era with cryptographic agility.

For potential future research directions, we aim to improve the forward security property in PQ−HASES. In fact, the forward security always incur an additional overhead that may be expensive, whether on the signer with additional computations and/or larger keys (e.g., XMSS^{MT}[18]) or at the verifier side with linear public keys (e.g., PQ−HASES). A potential research direction is to develop a forward security technique tailored for PQ digital signatures that provides efficiency for both the signer and the verifier. Additionally, we aim to further improve the security of HASES by using a set of distributed servers to construct the one-time public keys. This technique eliminates the high risk of relying on a central root of trust. Furthermore, HASES can be leveraged to generate the resource-intensive one-time commitments in multi-signatures, a process that traditionally involves multiple interactions during signing [60]. We also foresee that HASES can offer advantages to various applications (e.g., two-factor mobile authentication [55]), when used as a signature primitive.

ACKNOWLEDGMENT

This research is supported by the unrestricted gift from the Cisco Research Award (220159), and the NSF CAREER Award CNS-1917627.

REFERENCES

[1] Moayad Aloqaily, Ouns Bouachir, Fakhri Karray, Ismaeel Al Ridhawi, and Abdulmotaleb El Saddik. 2022. Integrating Digital Twin and Advanced Intelligent Technologies to Realize the Metaverse. *IEEE Consumer Electronics Mag.* (2022).

[2] Gaspard Anthoine, Jean-Guillaume Dumas, Mélanie de Jonghe, Aude Maignan, Clément Pernet, Michael Hanling, and Daniel S Roche. 2021. Dynamic proofs of retrievability with low server storage. In *30th USENIX Security Symposium (USENIX Security 21)*. 537–554.

[3] Giuseppe Ateniese, Roberto Di Pietro, Luigi V Mancini, and Gene Tsudik. 2008. Scalable and efficient provable data possession. In *Proc of the 4th international conference on Security and privacy in communication netowrks*. 1–10.

[4] Prithwi Bagchi, Basudeb Bera, Ashok Kumar Das, Sachin Shetty, Pandi Vijayakumar, and Marimuthu Karuppiah. 2023. Post Quantum Lattice-Based Secure Framework using Aggregate Signature for Ambient Intelligence Assisted Blockchain-Based IoT Applications. *IEEE Internet of Things Magazine* 6, 1 (2023), 52–58.

[5] Elaine Barker, Lily Chen, and Richard Davis. 2018. Recommendation for key-derivation methods in key-establishment schemes. *NIST Special Publication* 800 (2018), 56C.

[6] Rouzbeh Behnia and Attilla Altay Yavuz. 2021. Towards Practical Post-Quantum Signatures for Resource-Limited Internet of Things. In *Proceedings of the 37th Annual Computer Security Applications Conference (ACSAC '21)*. Association for Computing Machinery, New York, NY, USA, 119–130.

[7] Daniel J. Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, and Bo-Yin Yang. 2012. High-speed high-security signatures. *Journal of Cryptographic Engineering* 2, 2 (01 Sep 2012), 77–89.

[8] Daniel J Bernstein, Andreas Hülsing, Stefan Kölbl, Ruben Niederhagen, Joost Rijneveld, and Peter Schwabe. 2019. The SPHINCS+ signature framework. In *Proc. of the ACM SIGSAC conf. on computer and comm. security*. 2129–2146.

[9] Nina Bindel, Udyani Herath, Matthew McKague, and Douglas Stebila. 2017. Transitioning to a quantum-resistant public key infrastructure. In *Post-Quantum Cryptography: 8th International Workshop, PQCrypto 2017, Utrecht, The Netherlands, June 26-28, 2017, Proceedings 8*. Springer, 384–405.

[10] Alexandra Boldyreva, Craig Gentry, Adam O'Neill, and Dae Hyun Yum. 2007. Ordered multisignatures and identity-based sequential aggregate signatures, with applications to secure routing. In *Proceedings of the 14th ACM conference on Computer and communications security*. 276–285.

[11] Dan Boneh, Saba Eskandarian, and Ben Fisch. 2019. Post-quantum EPID signatures from symmetric primitives. In *Topics in Cryptology–CT-RSA 2019: The Cryptographers' Track at the RSA Conference 2019, San Francisco, CA, USA, March 4–8, 2019, Proceedings*. Springer, 251–271.

[12] Dan Boneh and Sam Kim. 2020. One-time and interactive aggregate signatures from lattices. *preprint* (2020).

[13] Dan Boneh, Ben Lynn, and Hovav Shacham. 2004. Short Signatures from the Weil Pairing. *J. Cryptol.* 17, 4 (2004).
[14] Katharina Boudgoust and Akira Takahashi. 2023. Sequential Half-Aggregation of Lattice-Based Signatures. *Cryptology ePrint Archive* (2023).
[15] Vinay Chamola, Alireza Jolfaei, Vaibhav Chanana, Prakhar Parashari, and Vikas Hassija. 2021. Information security in the post quantum era for 5G and beyond networks: Threats to existing cryptography, and post-quantum cryptography. *Computer Communications* 176 (2021), 99–118.
[16] Xue Chen, Shiyuan Xu, Yunhua He, Yu Cui, Jiahuan He, and Shang Gao. 2022. LFS-AS: lightweight forward secure aggregate signature for e-health scenarios. In *ICC 2022-IEEE Intern. Conf. on Communications*. IEEE, 1239–1244.
[17] Yanbo Chen and Yunlei Zhao. 2022. Half-aggregation of Schnorr signatures with tight reductions. In *European Symposium on Research in Computer Security*. Springer, 385–404.
[18] David A Cooper, Daniel C Apon, Quynh H Dang, Michael S Davidson, Morris J Dworkin, Carl A Miller, et al. 2020. Recommendation for stateful hash-based signature schemes. *NIST Special Publication* 800 (2020), 208.
[19] Victor Costan, Ilia Lebedev, and Srinivas Devadas. 2016. Sanctum: Minimal hardware extensions for strong software isolation. In *25th USENIX Security Symposium (USENIX Security 16)*. 857–874.
[20] Craig Costello and Patrick Longa. 2016. *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.
[21] Eric Crockett, Christian Paquin, and Douglas Stebila. 2019. Prototyping post-quantum and hybrid key exchange and authentication in TLS and SSH. *Cryptology ePrint Archive* (2019).
[22] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. 2021. Ascon v1. 2: Lightweight authenticated encryption and hashing. *Journal of Cryptology* 34 (2021), 1–42.
[23] Manu Drijvers, Sergey Gorbunov, Gregory Neven, and Hoeteck Wee. 2020. Pixel: Multi-signatures for Consensus.. In *USENIX Security Symposium*. 2093–2110.
[24] Léo Ducas, Eike Kiltz, Tancrede Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. 2018. Crystals-dilithium: A lattice-based digital signature scheme. *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2018), 238–268.
[25] Muhammad El-Hindi, Tobias Ziegler, Matthias Heinrich, Adrian Lutsch, Zheguang Zhao, and Carsten Binnig. 2022. Benchmarking the Second Generation of Intel SGX Hardware. In *Data Management on New Hardware*. 1–8.
[26] Abdulmotaleb El Saddik, Fedwa Laamarti, and Mohammad Alja'Afreh. 2021. The potential of digital twins. *IEEE Instrumentation & Measurement Magazine* 24, 3 (2021), 36–41.
[27] Thomas Espitau, Pierre-Alain Fouque, Benoît Gérard, and Mehdi Tibouchi. 2017. Side-channel attacks on BLISS lattice-based signatures: Exploiting branch tracing against strongswan and electromagnetic emanations in microcontrollers. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 1857–1874.
[28] Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Prest, Thomas Ricosset, Gregor Seiler, William Whyte, Zhenfei Zhang, et al. 2018. Falcon: Fast-Fourier lattice-based compact signatures over NTRU. *Submission to the NIST's post-quantum cryptography standardization process* 36, 5 (2018).
[29] Benjamin Glas, Jorge Guajardo, Hamit Hacioglu, Markus Ihle, Karsten Wehefritz, and Attila A. Yavuz. 2012. Signal-based Automotive Communication Security and Its Interplay with Safety Requirements. ESCAR, Embedded Security in Cars Conference, Germany, November 2012.
[30] Andreas Huelsing, Denis Butin, Stefan-Lukas Gazdag, Joost Rijneveld, and Aziz Mohaisen. 2018. XMSS: eXtended Merkle Signature Scheme. RFC 8391. https://doi.org/10.17487/RFC8391
[31] David Joseph, Rafael Misoczki, Marc Manzano, Joe Tricot, Fernando Dominguez Pinuaga, Olivier Lacombe, Stefan Leichenauer, Jack Hidary, Phil Venables, and Royal Hansen. 2022. Transitioning organizations to post-quantum cryptography. *Nature* 605, 7909 (2022), 237–243.
[32] Jonathan Katz and Yehuda Lindell. 2020. *Introduction to modern cryptography*. CRC press.
[33] Fan Lang, Wei Wang, Lingjia Meng, Qiongxiao Wang, Jingqiang Lin, and Li Song. 2021. Informer: Protecting intel sgx from cross-core side channel threats. In *Intern. Conf. on Information and Communications Security*. 310–328.
[34] Tian Li, Huaqun Wang, Debiao He, and Jia Yu. 2020. Permissioned blockchain-based anonymous and traceable aggregate signature scheme for Industrial Internet of Things. *IEEE Internet of Things Journal* 8, 10 (2020), 8387–8398.
[35] T. Malkin, D. Micciancio, and S. K. Miner. 2002. Efficient Generic Forward-Secure Signatures with an Unbounded Number Of Time Periods. In *Proc. of the 21th International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT '02)*. Springer-Verlag, 400–417.
[36] Hongzi Mao, Mohammad Alizadeh, Ishai Menache, and Srikanth Kandula. 2016. Resource management with deep reinforcement learning. In *Proceedings of the 15th ACM workshop on hot topics in networks*. 50–56.
[37] NIST. [n. d.]. PQC Standardization Process: Announcing Four Candidates to be Standardized, Plus Fourth Round Candidates. https://csrc.nist.gov/News/2022/pqc-candidates-to-be-standardized-and-round-4. Accessed: July 14, 2022.

[38] Saif E. Nouma, , and Attila A. Yavuz. 2023. Post-Quantum Forward-Secure Signatures with Hardware-Support for Internet of Things *(ICC 2023-IEEE International Conference on Communications)*. IEEE, 4540–4545.
[39] Saif E Nouma and Attila A Yavuz. 2023. Practical Cryptographic Forensic Tools for Lightweight Internet of Things and Cold Storage Systems. In *Proc. of the 8th ACM/IEEE Conf. on Internet of Things Design and Implementation*. 340–353.
[40] David Ott, Christopher Peikert, et al. 2019. Identifying research challenges in post quantum cryptography migration and cryptographic agility. *arXiv preprint arXiv:1909.07353* (2019).
[41] Wenyi Ouyang, Qiongxiao Wang, Wei Wang, Jingqiang Lin, and Yaxi He. 2021. SCB: Flexible and Efficient Asymmetric Computations Utilizing Symmetric Cryptosystems Implemented with Intel SGX. In *2021 IEEE International Performance, Computing, and Communications Conference (IPCCC)*. 1–8.
[42] Muslum Ozgur Ozmen, Rouzbeh Behnia, and Attila A. Yavuz. 2019. Energy-Aware Digital Signatures for Embedded Medical Devices. In *7th IEEE Conf. on Communications and Network Security (CNS), Washington, D.C., USA, June*.
[43] Sebastian Paul and Patrik Scheible. 2020. Towards post-quantum security for cyber-physical systems: Integrating PQC into industrial M2M communication. In *Computer Security–ESORICS 2020: 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, September 14–18, 2020, Proceedings, Part II 25*. Springer, 295–316.
[44] D. Pointcheval and J. Stern. 1996. Security proofs for signature schemes. In *Proc. of the 15th International Conference on the Theory and Application of Cryptographic Techniques (EUROCRYPT '96)*. Springer-Verlag, 387–398.
[45] Yousef Qassim, Mario Edgardo Magaña, and Attila Yavuz. 2017. Post-quantum hybrid security mechanism for MIMO systems. In *Computing, Networking and Communications (ICNC), 2017 International Conference on*. IEEE, 684–689.
[46] Leonid Reyzin and Natan Reyzin. 2002. Better than BiBa: Short One-Time Signatures with Fast Signing and Verifying. In *Information Security and Privacy: 7th Australasian Conference*. 144–153.
[47] Efe U. A. Seyitoglu, Attila A. Yavuz, and Muslum O. Ozmen. 2020. Compact and Resilient Cryptographic Tools for Digital Forensics. In *2020 IEEE Conference on Communications and Network Security (CNS)*. 1–9.
[48] Surbhi Shaw and Ratna Dutta. 2022. Post-quantum secure identity-based signature achieving forward secrecy. *Journal of Information Security and Applications* 69 (2022), 103275.
[49] Wei Shengli. 2021. Is human digital twin possible? *Computer Methods and Programs in Biomedicine Update* 1 (2021), 100014.
[50] Peter W. Shor. 1999. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM Rev.* 41, 2 (1999), 303–332.
[51] Rangana De Silva, Iranga Navaratna, Malitha Kumarasiri, Janaka Alawatugoda, and Chuah Chai Wen. 2022. On power analysis attacks against hardware stream ciphers. *Intern. J. of Information and Computer Security* 17, 1-2 (2022), 21–35.
[52] Thokozani F. Vallent, Damien Hanyurwimfura, and Chomora Mikeka. 2021. Efficient certificateless aggregate signature scheme with conditional privacy-preservation for vehicular adhoc networks enhanced smart grid system. *Sensors* (2021).
[53] Cong Wang, Sherman SM Chow, Qian Wang, Kui Ren, and Wenjing Lou. 2011. Privacy-preserving public auditing for secure cloud storage. *IEEE transactions on computers* 62, 2 (2011), 362–375.
[54] Ding Wang, Debiao He, Ping Wang, and Chao-Hsien Chu. 2014. Anonymous two-factor authentication in distributed systems: Certain goals are beyond attainment. *IEEE TDSC* 12, 4 (2014), 428–442.
[55] Ding Wang and Ping Wang. 2016. Two birds with one stone: Two-factor authentication with security beyond conventional bound. *IEEE transactions on dependable and secure computing* 15, 4 (2016), 708–722.
[56] Qingxuan Wang, Ding Wang, Chi Cheng, and Debiao He. 2021. Quantum2fa: efficient quantum-resistant two-factor authentication scheme for mobile devices. *IEEE Transactions on Dependable and Secure Computing* (2021).
[57] Attila A. Yavuz. 2018. Immutable Authentication and Integrity Schemes for Outsourced Databases. *IEEE Trans. Dependable Sec. Comput.* 15, 1 (2018), 69–82.
[58] Attila A Yavuz and Rouzbeh Behnia. 2022. FROG: Forward-Secure Post-Quantum Signature. *arXiv preprint arXiv:2205.07112* (2022).
[59] Attila A. Yavuz, Peng Ning, and Michael K. Reiter. 2012. BAF and FI-BAF: Efficient and Publicly Verifiable Cryptographic Schemes for Secure Logging in Resource-Constrained Systems. *ACM Trans on Information System Security* 15, 2 (2012), 28 pages.
[60] Attila Altay Yavuz and Saif Nouma. 2023. Hardware supported authentication and signatures for wireless, distributed and blockchain systems. US Patent App. 18/188,749.
[61] Attila A Yavuz, Saif E Nouma, Thang Hoang, Duncan Earl, and Scott Packard. 2022. Distributed Cyber-infrastructures and Artificial Intelligence in Hybrid Post-Quantum Era. In *2022 IEEE 4th International Conference on Trust, Privacy and Security in Intelligent Systems, and Applications (TPS-ISA)*. IEEE, 29–38.
[62] Haojin Zhu, Xiaodong Lin, Rongxing Lu, Yanfei Fan, and Xuemin Shen. 2009. Smart: A secure multilayer credit-based incentive scheme for delay-tolerant networks. *IEEE transactions on vehicular technology* 58, 8 (2009), 4628–4639.

APPENDIX A

Our schemes rely on HORS [46] and Schnorr [20] digital signatures that are defined as below.

Definition 7.1. The Hash to Obtain Random Subset (HORS) signature [46] is a tuple of three algorithms $(\mathtt{Kg}, \mathtt{Sig}, \mathtt{Ver})$ defined as follows:

- $(sk, PK, I) \leftarrow \mathtt{HORS.Kg}(1^\kappa)$: Given the security parameter κ, it first generates $I \leftarrow (l, k, t)$, and then t random l-bit strings $\{s_i\}_{i=1}^t$ and $\{v_i \leftarrow f(s_i)\}_{i=1}^t$. It sets $sk \leftarrow \{s_i\}_{i=1}^t$ and $PK \leftarrow \{v_i\}_{i=1}^t$.
- $\sigma \leftarrow \mathtt{HORS.Sig}(sk, M)$: Given (sk, M), it computes $h \leftarrow H_0(M)$, splits it as $\{h_j\}_{j=1}^k$ where $|h_j| = \log t$ and interprets them as integers $\{i_j\}_{j=1}^k$. It sets $\sigma \leftarrow \{s_{i_j}\}_{j=1}^k$.
- $b \leftarrow \mathtt{HORS.Ver}(PK, M, \sigma)$: Given PK, M, and σ, it computes $\{i_j\}_{j=1}^k$ as in $\mathtt{HORS.Sig}(.)$ and checks if $f(s'_j) = v_{i_j}$, $j = 1, \ldots, k$, returns $b = 1$, else $b = 0$.

Definition 7.2. The Schnorr signature [20] is a tuple of three algorithms defined as follows:

- $(y, Y, I) \leftarrow \mathtt{Schnorr.Kg}(1^\kappa)$: Given the security parameter κ, it first generates large primes q and $p > q$ such that $q|(p-1)$. Select a generator α of the subgroup G of order q in $\mathbb{Z}_p^*$.
- $\sigma \leftarrow \mathtt{Schnorr.Sig}(y, M)$: Given (y, M), it generates private/public commitment $(r \xleftarrow{\$} \mathbb{Z}_q^*, R \leftarrow \alpha^r \mod p)$. It computes an ephemeral key $e \leftarrow H_0(M\|R)$ and the signature $s \leftarrow r - e \cdot y$. It sets $\sigma \leftarrow \langle s, e \rangle$.
- $b \leftarrow \mathtt{Schnorr.Ver}(Y, M, \sigma)$: Given Y, M, and σ, it computes $R' \leftarrow Y^e \cdot \alpha^s \mod p$. It checks if $e = H(M\|R)$, then it returns $b = 1$, otherwise $b = 0$.

The Discrete Logarithm Problem (DLP) is defined as below:

Definition 7.3. Let $\mathbb{G}$ be a cyclic group of order q, let α be a generator of $\mathbb{G}$, and let DLP attacker $\mathcal{A}$ be an algorithm that returns an integer in $\mathbb{Z}_q$. We consider the following experiment:

Experiment $Expt^{DL}_{\mathbb{G},\alpha}(\mathcal{A})$:

$b \xleftarrow{\$} \mathbb{Z}_q^*$, $B \leftarrow \alpha^b \mod q$, $b' \leftarrow \mathcal{A}(B)$,

If $\alpha^{b'} \mod p = B$ then return 1, else return 0

The DL-advantage of $\mathcal{A}$ in this experiment is defined as: $Adv^{DL}_G(\mathcal{A}) = Pr[Expt^{DL}_{\mathbb{G},\alpha}(\mathcal{A}) = 1]$

The *DL* advantage of $(\mathbb{G}, \alpha)$ in this experiment is defined as follows: $Adv^{DL}_G(t) = \max_{\mathcal{A}}\{Adv^{DL}_G(\mathcal{A})\}$, where the maximum is over all $\mathcal{A}$ having time complexity t.

We implemented our schemes in the EC domain due to the small key sizes. Specifically, we used the curve of Ed25519 [7] that offers efficient arithmetics. The security of Ed25519 relies on Elliptic Curve Discrete Logarithm Problem (ECDLP), which is the EC variant of DLP in Definition 7.3.

APPENDIX B

We provide the security proof of PQ−HASES scheme as below.

THEOREM 5.1. $Adv^{F-EU-CMA}_{PQ-HASES}(t, q_s, q'_s, 1) \leq J \cdot Adv^{EU-CMA}_{HORS}(t', q_s, q'_s)$, *where* $O(t') = O(t) + J \cdot H$.

Proof: $\mathcal{F}$ is given the challenge public key PK', where $(sk', PK', I) \leftarrow \mathtt{HORS.Kg}(1^\kappa)$.

Algorithm $\mathcal{F}^{RO(.)(\cdot),\mathtt{HORS}_{sk'}(\cdot),\mathtt{Break-In(j)}}(PK')$: $\mathcal{F}$ is run per Def. 3.1 as follows.

• *Setup:* $\mathcal{F}$ maintains $\mathcal{LM}$, and $\mathcal{LS}$ to track the query results in the experiment duration. $\mathcal{LM}$ is a list of queried messages to the signing oracle $\mathtt{PQ-HASES.Sig}_{sk'}$. $\mathcal{LS}$ is a signature list to record answers given by $\mathtt{PQ-HASES.Sig}_{sk'}$. $\mathcal{LH}$ is a hash list in form of pairs $\{(M_l, k) : h_l\}$, where M_l is the l^{th} data item queried to $RO(.)$

and h_l is its associated $RO(.)$ answer. $k = 0$ denotes the selection of hash function H or $k = 1$ in case of PRF function. $\mathcal{A}$ selects an ID_n and gives it to $\mathcal{F}$. If $ID_n \notin \vec{ID}$, then $\mathcal{F}$ *aborts*, else continues. The index n, for the user ID_n, is omitted for brevity. $\mathcal{F}$ uses a function *H-Sim* to model H as a random oracle $RO(.)$: If $\exists(M,k) : \mathcal{LH}[M,k] = h$ then *H-Sim* returns h. Else, it returns $h \leftarrow \{0,1\}^\kappa$ as the answer for H_k, insert a new pair $\mathcal{LH}(M,k) \leftarrow h$ and update $l \leftarrow l+1$. $\mathcal{F}$ selects a target forgery index $w \in [1,J]$ and sets $C_w \leftarrow PK'$. $\mathcal{F}$ generates $msk \xleftarrow{\$} \{0,1\}^\kappa$, and computes $\{sk_j\}_{j=1,j\neq w}^{J}$ and $\{C_j, \sigma_{C_j}\}_{j=1,j\neq w}^{J}$ as in PQ-HASES.Kg and PQ-HASES.ComConstr via *msk* and *csk*, where H and PRF calls are simulated with *H-Sim*.

• *Queries:* $\mathcal{F}$ handles $\mathcal{A}$'s queries as follows.

(1) $RO(.)$: $\mathcal{A}$ queries $RO(.)$ on q'_s messages of her choice. When $\mathcal{A}$ queries M for a hash function $H_{k=0,1}$, $\mathcal{F}$ returns $h \leftarrow$ *H-Sim*$(M,k,l,\mathcal{LH})$.

(2) $Sig_{sk_j}(.)$: If $\mathcal{A}$ queries $\mathcal{F}$ on M_w then $\mathcal{F}$ returns $\sigma_j \leftarrow \text{HORS}_{sk'}(M_j)$ by querying $\text{HORS}_{sk'}(.)$ signing oracle. Otherwise, $\mathcal{F}$ returns $\sigma_j \leftarrow$ PQ-HASES.Sig$(sk_j \in \mathcal{LH}, M_j)$ (sk_j is a PRF output, and therefore it is available in $\mathcal{LH}$).

(3) $ComConstr_{msk}(.)$: If $\mathcal{A}$ queries $\mathcal{F}$ on w$^{\text{th}}$ commitment, then $\mathcal{F}$ returns $C_w = PK'$. Else, $\mathcal{F}$ returns C_j from $\mathcal{LH}$ that is computed via PQ-HASES.ComConstr algorithm under *msk*.

(4) $Break\text{-}In(j)$: If $j = J$ then $\mathcal{F}$ rejects the query (all private keys were used) and proceed to the forgery phase. If $1 < j \leq w$ then $\mathcal{F}$ *aborts*, else it returns $sk_j \in \mathcal{LH}$ and ends the experiment.

• *Forgery:* $\mathcal{A}$ outputs a forgery (M^*, σ^*) on PK^*. $\mathcal{F}$ wins the experiments if $\mathcal{A}$ wins the experiments by producing a valid forgery on PK_w. That is, $\mathcal{F}$ returns 1 if $PK^* = PK_w$, M^* was not queried to $\text{Sig}_{sk_j}(.)$, and $1 =$ HORS.Ver(PK^*, M^*, σ^*). Otherwise, $\mathcal{F}$ returns 0 and *aborts*.

• *Success Probability and Indistinguishability Argument*: Assume that $\mathcal{A}$ wins F-EU-CMA experiment against PQ-HASES with the probability $Adv^{\text{F-EU-CMA}}_{\text{PQ-HASES}}(t, q_s, q'_s, 1)$. $\mathcal{F}$ wins the EU-CMA experiments against HORS if and only if $\mathcal{A}$ produces a forgery on the challenge public key PK_w and does not abort during the experiment. Since $w \in [1,J]$ is randomly selected, the success probability of $\mathcal{A}$ can be bounded to that of $\mathcal{F}$ as: $Adv^{\text{F-EU-CMA}}_{\text{PQ-HASES}}(t, q_s, q'_s, 1) \leq J \cdot Adv^{\text{EU-CMA}}_{HORS}(t', q_s, q'_s)$

$\mathcal{F}$'s transcripts in all lists are identical to the real execution except that PK_w is replaced with HORS public key PK'. Hence, $sk_w = sk'$ is not part of hash chain generated from sk_1 or *msk*. As in HORS, H is a random oracle. Thus, w^{th} element of lists in $\mathcal{A}_\mathcal{R}$ and $\mathcal{A}_\mathcal{S}$ have identical distributions. ■

We give the security proof of LA-HASES as below.

THEOREM 5.2. *$Adv^{A\text{-}EU\text{-}CMA}_{LA\text{-}HASES}(t, q_s, q'_s) \leq Adv^{DL}_{G,\alpha}(t')$, where $t' = O(t) + O(\kappa^3 + L \cdot RNG)$.*

Proof: Let $\mathcal{A}$ be a LA-HASES attacker. We construct a *DL-attacker* $\mathcal{F}$ that uses $\mathcal{A}$ as a sub-routine. That is, we set ($b \xleftarrow{\$} \mathbb{Z}^*_q$, $B \leftarrow \alpha^b \bmod p$) as in Def. 7.3 and then run the simulator $\mathcal{F}$ by Def. 3.2 (i.e., A-EU-CMA) as follows:

Algorithm $\mathcal{F}(B)$: $\mathcal{F}$ is run per Definition 3.2.

Setup: $\mathcal{F}$ maintains $\mathcal{LH}$, $\mathcal{LM}$, $\mathcal{LS}$, $\mathcal{LC}$ and $\mathcal{LR}$ to keep track of query results in the duration of the experiments. $\mathcal{LH}$ is a hash list in form of pairs $\{(M_l, k) : h_l\}$, where (M_l, h_l) represents the l^{th} data item queried to $RO(.)$ and its corresponding $RO(.)$ answer, respectively. $k = 0$ and $k = 1$ denotes the selection of hash function H and PRF, respectively. $\mathcal{LM}$ is a list containing vectors of messages, each of its elements $\mathcal{LM}[j]$ is a message vector $\overrightarrow{M}_j = \{m^\ell_j\}_{\ell=1}^{L}$ (i.e., the j^{th} batch query to LA-HASES.ASig$_{sk}$). $\mathcal{LS}$ is a signature list, used to record answers given by LA-HASES.ASig$_{sk}$. $\mathcal{LR}$ is a list, used to record the randomly generated variables. $\mathcal{LR}[0,0,0]$ and $\mathcal{LR}[1,j,\ell]$ refer to the simulated private key z (generated by $RO(.)$) and s^ℓ_j during the message element index ℓ of the batch query j, respectively.

- $\mathcal{F}$ sets the public key and the system-wide parameters as follows: $Y \leftarrow B, z \xleftarrow{\$} \mathbb{Z}_q^*$ and add to $\mathcal{LR}$ as $\mathcal{LR}[0,0,0] \leftarrow z$. Set $I \leftarrow (p, q, \alpha, J, L)$, and initialize the counters $(l \leftarrow 0, j \leftarrow 1)$.
- $\mathcal{A}$ selects an ID and gives it to $\mathcal{F}$. If $ID \notin \vec{ID}$, $\mathcal{F}$ aborts, else continues (user index n is omitted).
- $h \leftarrow H\text{-}Sim(M, k, l, \mathcal{LH})$: $\mathcal{F}$ uses a function $H\text{-}Sim$ that works as a random oracle, $RO(.)$. Note that the hash and PRF functions $H_{k=0,1}$ are modeled as random oracles. If $\exists (M,k) : \mathcal{LH}[M,k] = h$ then $H\text{-}Sim$ returns h. Otherwise, return $h \xleftarrow{\$} \mathbb{Z}_q^*$ as the answer for H_k, insert a new pair $\mathcal{LH}(M,k) \leftarrow h$ and update $l \leftarrow l+1$.

Execute $\mathcal{A}^{RO(.),\texttt{LA-HASES.ASig}_{sk}(\cdot),\texttt{LA-HASES.ComConstr}(\cdot)}(PK)$:

• *Queries:* $\mathcal{F}$ handles $\mathcal{A}$'s queries as follows:

(1) $RO(.)$: $\mathcal{A}$ queries $RO(.)$ on q_s' messages of her choice. When $\mathcal{A}$ queries M for a hash function $H_{k=0,1}$, $\mathcal{F}$ returns $h \leftarrow H\text{-}Sim(M, k, l, \mathcal{LH})$.

(2) $\texttt{Sig}_{sk}(.)$: $\mathcal{A}$ queries the LA-HASES.ASig(.) oracle on J data batches of her choice $\vec{M}_j = \{m_j^\ell\}_{\ell=1}^L, \forall j \in [1, J]$. If $j > J$, $\mathcal{F}$ rejects the query (exceed limit), else it continues as follows:
   a) Compute $z \leftarrow \mathcal{LR}[0,0,0]$, $x_j \leftarrow H\text{-}Sim(z \| j, 0, l, \mathcal{LH})$, and set $s_j^{1,0} \leftarrow 0$
   b) **for** $\ell = 1, \ldots, L$ **do**
      i) Compute $x_j^\ell \leftarrow H\text{-}Sim(x_j \| \ell, 0, l, \mathcal{LH})$ and $e_j^\ell \leftarrow H\text{-}Sim(x_j^\ell, 2, l, \mathcal{LH})$
      ii) If $(m_j^\ell \| x_j^\ell, 2) \in \mathcal{LH}$ then $\mathcal{F}$ aborts. Otherwise, it computes $H\text{-}Sim(m_j^\ell \| x_j^\ell, 2, l, \mathcal{LH})$
      iii) If $\exists (j, \ell) : \mathcal{LR}[1, j, \ell] = s_j^\ell$ then $s_j^\ell \leftarrow \mathcal{LR}[2, j, \ell]$. Otherwise, $s_j^\ell \xleftarrow{\$} \mathbb{Z}_q^*$ and insert it into $\mathcal{LR}$ as $\mathcal{LR}[1, j, \ell] \leftarrow s_j^\ell$. Compute $s_j^{1,\ell} \leftarrow s_j^{1,\ell-1} + s_j^\ell \mod q$
   c) $\mathcal{F}$ sets $\sigma_j^{1,L} \leftarrow \langle s_j^{1,L}, x_j, ID, j \rangle$, insert $(\vec{M}_j, \sigma_j^{1,L})$ to $(\mathcal{LM}, \mathcal{LS})$ and increment $j \leftarrow j+1$

(3) $\texttt{ComConstr}_{msk}(.)$: $\mathcal{A}$ queries LA-HASES.ComConstr oracle on user ID and counter j of her choice. If $j \notin [1, J]$ or $ID \notin \vec{ID}$, $\mathcal{F}$ reject the query, else it continues as follows:
   a) $e_j^{1,0} \leftarrow 0$ and $s_j^{1,0} \leftarrow 0$. Retrieve $z \leftarrow \mathcal{LR}[0,0,0]$ and Compute $x_j \leftarrow H\text{-}Sim[z \| j, 0, l, \mathcal{LH}]$
   b) **for** $\ell = 1, \ldots, L$ **do**
      i) Compute $x_j^\ell \leftarrow H\text{-}Sim(x_j \| \ell, 0, l, \mathcal{LH})$ and $e_j^\ell \leftarrow H\text{-}Sim(x_j^\ell, 2, l, \mathcal{LH})$
      ii) If $\exists (j, \ell) : \mathcal{LR}[2, j, \ell] = s_j^\ell$ then $s_j^\ell \leftarrow \mathcal{LR}[1, j, \ell]$. Otherwise, generate $s_j^\ell \xleftarrow{\$} \mathbb{Z}_q^*$ and insert $\mathcal{LR}[1, j, \ell] \leftarrow s_j^\ell$
   c) Set $R_j^{1,L} \leftarrow Y^{\sum_{\ell=1}^L e_j^\ell} \cdot \alpha^{\sum_{\ell=1}^L s_j^\ell} \mod p$, return and insert $C_j^{1,L} \leftarrow R_j^{1,L}$ to $\mathcal{LC}$ as $\mathcal{LC}[j] \leftarrow C_j^{1,L}$.

- Forgery of $\mathcal{A}$: Eventually, $\mathcal{A}$ outputs forgery on PK as $(\vec{M}_j^*, \sigma^{*1,L}_j)$, for $j \in [1, J]$, where $\vec{M}_j^* = \{m^{*1}_j, \ldots, m^{*L}_j\}$ and $\sigma^{*1,L}_j = \langle s^{*1,L}_j, x_j^*, ID \rangle$. By definition 3.2, $\mathcal{A}$ wins A-EU-CMA experiment for LA-HASES if LA-HASES.AVer$(\langle PK, C_j^{1,L} \rangle, \vec{M}_j^*, \sigma^{*1,L}_j) = 1$ and $\vec{M}_j^* \notin \mathcal{LM}$ hold, where $C_j^{1,L} \leftarrow$ LA-HASES.$\texttt{ComConstr}_{msk}(ID, j)$. If these conditions hold, $\mathcal{A}$ returns 1, otherwise 0.

- Forgery of $\mathcal{F}$: If $\mathcal{A}$ loses in the A-EU-CMA experiment for LA-HASES, $\mathcal{F}$ also loses in the DL experiment, and thus $\mathcal{F}$ *aborts* and returns 0. Else, if $\vec{M}_j^* \in \mathcal{LM}$ then $\mathcal{F}$ *aborts* and returns 0 (i.e., $\mathcal{A}$ wins the experiment without querying $RO(.)$ oracle). Otherwise, $\mathcal{F}$ continues as follows:
  $R_j^{1,L} \equiv Y^{e_j^{1,L}} \cdot \alpha^{s_j^{1,L}} \mod p$ holds for the aggregated variables $(R_j^{1,L}, e_j^{1,L}, s_j^{1,L})$. That is, $\mathcal{F}$ retrieves $z \leftarrow \mathcal{LR}[0,0,0]$ and computes $x_j \leftarrow H\text{-}Sim(z \| j, 2, l, \mathcal{LH})$. Then, it computes $e_j^{1,L} \leftarrow \sum_{\ell=1}^L H\text{-}Sim(x_j^\ell, 2, l, \mathcal{LH})$ where $x_j^\ell = H\text{-}Sim(x_j \| \ell, 0, l, \mathcal{LH}), \forall \ell \in [1, L]$. Also, $\mathcal{F}$ computes $s_j^{1,L} \leftarrow \sum_{\ell=1}^L \mathcal{LR}[1, j, \ell]$.

Moreover, $\texttt{LA-HASES.AVer}(\vec{M}^*_j, \sigma^{*1,L}_j) = 1$ holds, and therefore $R^{1,L}_j \equiv Y^{e^{*1,L}_j} \cdot \alpha^{s^{*1,L}_j} \bmod p$ also holds. Therefore, $\mathcal{F}$ computes $x^{*\ell}_j \leftarrow H\text{-}Sim(x^*_j \| \ell, 0, l, \mathcal{LH}), \forall \ell \in [1, L]$ and calculates $e^{*1,L}_j \leftarrow \sum_{\ell=1}^{L} H\text{-}Sim(m^{*\ell}_j \| x^{*\ell}_j, 2, l, \mathcal{LH})$. Thus, the following equations hold:

$$R^{1,L}_j \equiv Y^{e^{1,L}_j} \cdot \alpha^{s^{1,L}_j} \bmod p, \;\; R^{1,L}_j \equiv Y^{e^{*1,L}_j} \cdot \alpha^{s^{*1,L}_j} \bmod p,$$

$\mathcal{F}$ then extracts $y' = b$ by solving the below modular linear equations (note that only unknowns are y' and $r^{1,L}_j$), where $Y = B$ as defined in the public key simulation:

$$r^{1,L}_j \equiv y' \cdot e^{1,L}_j + s^{1,L}_j \bmod q\,, \;\; r^{1,L}_j \equiv y' \cdot e^{*1,L}_j + s^{*1,L}_j \bmod q,$$

$B' \equiv \alpha^b \bmod p$ holds as $\mathcal{A}$'s forgery is valid and non-trivial on $B' = B$. By Def. 7.3, $\mathcal{F}$ wins the *DL-experiment*.

Execution Time Analysis: The runtime of $\mathcal{F}$ is that of $\mathcal{A}$ plus the time to respond to $RO(.)$ queries. We denote the approximate cost of drawing a random number and modular exponentiation with $O(\kappa)$ and $O(\kappa^3)$, respectively. In the setup phase, $\mathcal{F}$ draws random numbers with a negligible cost. In the query phase, $\mathcal{F}$ performs $O(3 \cdot L \cdot \kappa)$ to handle $\texttt{ASig}_{sk}(.)$ oracle queries. To answer $\texttt{ComConstr}_{msk}(.)$ queries, it performs $O(2 \cdot L \cdot \kappa)$ (seed and ephemeral key derivation), and $O(2 \cdot \kappa^3)$ for two modular exponentiations. The overall cost of query phase is bounded as $O(\kappa^3)$. Therefore, the approximate total running time of $\mathcal{F}$ is $t' = O(t) + O(\kappa^3)$.

Success Probability Analysis: $\mathcal{F}$ succeeds if all below events occur.

- $\overline{E1}$: $\mathcal{F}$ does not abort during the query phase.
- $E2$: $\mathcal{A}$ wins the A-EU-CMA experiment for LA-HASES.
- $\overline{E3}$: $\mathcal{F}$ does not abort after $\mathcal{A}$'s forgery.
- Win: $\mathcal{F}$ wins the A-EU-CMA experiment for *DL-experiment*.
- $Pr[Win] = Pr[\overline{E1}] \cdot Pr[E2|\overline{E1}] \cdot Pr[\overline{E3}|\overline{E1} \wedge E2]$

• *The probability that event $\overline{E1}$ occurs*: During the query phase, $\mathcal{F}$ aborts if $(m^\ell_j \| x^\ell_j, k = 0) \in \mathcal{LH}, \forall \ell \in [1, L]$ holds, *before* $\mathcal{F}$ inserts $(m^\ell_j \,\|\, x^\ell_j, k = 0)$ into $\mathcal{LH}$. This occurs if $\mathcal{A}$ guesses x^ℓ_j (before it is released) and then queries $(m^\ell_j \,\|\, x^\ell_j)$ to $RO(.)$ *before* querying it to $\texttt{LA-HASES.ASig}(.)$. The probability that this occurs is $\frac{1}{2^\kappa}$, which is negligible in terms of κ. Hence, $Pr[\overline{E1}] = (1 - \frac{1}{2^\kappa}) \approx 1$.

• *The probability that event E2 occurs*: If $\mathcal{F}$ does not abort, $\mathcal{A}$ also does not abort since the $\mathcal{A}$'s simulated view is *indistinguishable* from $\mathcal{A}$'s real view. Thus, $Pr[E2|\overline{E1}] = Adv^{\texttt{A-EU-CMA}}_{\texttt{LA-HASES}}(t, q_s, q'_s)$.

• *The probability that event $\overline{E3}$ occurs*: $\mathcal{F}$ does not abort if the following conditions are satisfied: (i) $\mathcal{A}$ wins the A-EU-CMA experiment for LA-HASES on a message M^* by querying it to $RO(.)$. The probability that $\mathcal{A}$ wins without querying M^* to $RO(.)$ is as difficult as a random guess. (ii) After $\mathcal{F}$ extracts $y' = b$ by solving modular linear equations, the probability that $Y' \not\equiv \alpha^{y'} \bmod p$ is negligible in terms κ, since $(Y = B) \in PK$ and $\texttt{LA-HASES.AVer}(PK, M^*, \sigma^*) = 1$. Hence, $Pr[\overline{E3}|\overline{E1} \wedge E2] = Adv^{\texttt{A-EU-CMA}}_{\texttt{LA-HASES}}(t, q_s, q'_s)$. Omitting the terms that are negligible in terms of κ, the upper bound on *A-EU-CMA-advantage of LA-HASES* is: $Adv^{\texttt{A-EU-CMA}}_{\texttt{LA-HASES}}(t, q_s, q'_s) \leq Adv^{DL}_{G,\alpha}(t')$.

Indistinguishability Argument: The real-view of $\overrightarrow{A}_{real}$ is comprised of the public key PK, system-wide parameters $\overline{I}$, the answers of $\texttt{LA-HASES.ASig}_{sk}(.)$ and $\texttt{LA-HASES.ComConstr}_{msk}(.)$ (recorded in $\mathcal{LS}$ and $\mathcal{LC}$ by $\mathcal{F}$, respectively), and the answer of $RO(.)$ (recorded in $\mathcal{LH}$ and $\mathcal{LR}$ by $\mathcal{F}$). All these values are generated by LA-HASES algorithms as in the real system, where $sk = y$ serves as the initial randomness. The joint probability distribution of $\overrightarrow{A}_{real}$ is random uniform as that of sk.

The simulated view of $\mathcal{A}$ is as $\overrightarrow{A}_{sim}$, and it is equivalent to $\overrightarrow{A}_{real}$ except that in the simulation, the signature components $\{s_j^{\ell}\}_{j\in[1,J],\ell\in[1,L]}$ are randomly selected from $\mathbb{Z}_q^*$. The private key is simulated as $y = z$, from which we derive the seeds $\{x_j\}_{j\in[1,J]}$ and the ephemeral keys $\{e_j^{1,L}\}_{j\in[1,J]}$. This dictates the selection of the aggregate signatures $\{s_j^{1,L}\}_{j\in[1,J]}$ and the aggregate commitments $\{R_j^{1,L}\}_{j\in[1,J]}$ as random via $\texttt{ASig}_{sk}(.)$ and $\texttt{ComConstr}_{msk}(.)$ oracles, repectively. Note that the joint probability distribution of these values is also randomly and uniformly distributed and is identical to the original signatures and hash outputs in $\overrightarrow{A}_{real}$ (since $H_{k=0,1}$ are modeled as $RO(.)$ via *H-Sim*). ■

On the Forking Lemma for Special-Condition in Schnorr-like Signatures: Schnorr [20] employs the forking lemma in its proof with a tight bound proven [44]. Nonetheless, one-time Schnorr-like signatures (e.g., [59]) offer an alternative approach that eliminates the need for the forking lemma, but with the cost of pre-determined number of signing and linear public key size (or verifier transmission [42]). This trade-off arises because the commitment (i.e., R) is pre-fixed in PK, meaning that the adversary $\mathcal{A}$ must forge using same commitment without the need to rewind the tape [44]. Inline with these previous works, LA−HASES generates randomness during signature generation, derived from initial randomness established during key generation. LA−HASES introduces the concept of CCO to provide verifiers with one-time commitments in offline or on-demand. When an adversary $\mathcal{A}$ attempts forgery, they are supplied with aggregate commitments from $\texttt{ComConstr}_{msk}(.)$. Consequently, LA−HASES eliminates the need for the forking lemma, resulting in a more robust security guarantee.

We provide the security proof of HY−HASES as below.

THEOREM 5.3. $Adv^{H-EU-CMA}_{HY-HASES}(t, q_s, q'_s) = \min\{Adv^{EU-CMA}_{PQ-HASES}(t, q_s, q'_s, 1)\ ,\ Adv^{EU-CMA}_{PQ-HASES}(t, q_s, q'_s)\}$.

Proof: Let $\mathcal{A}$ finds a forgery against HY−HASES with a nested combination of PQ−HASES and LA−HASES by outputting $q_s + 1$ valid HY−HASES signatures under distinct message batches. We can then construct an algorithm $\mathcal{A}_{LA}$ that finds a forgery in LA−HASES. $\mathcal{A}_{LA}$ interacts with $\mathcal{F}_{LA}$ for LA−HASES which provides a public key PK_{LA}. $\mathcal{A}_{LA}$ derives a user's key pair (sk_{PQ}, PK_{PQ}) by first generating $(msk_{PQ}, \vec{sk}_{PQ}, I) \leftarrow \texttt{PQ−HASES.Kg}(1^{\kappa}, J, L)$. It sets the user's public key of HY−HASES to be $PK_{HY} \leftarrow \langle PK_{LA}, PK_{PQ}\rangle$. When $\mathcal{A}$ asks for a batch of messages $\vec{M}_j = \{m_j^{\ell}\}_{\ell=1}^{L}$ to be signed using HY−HASES, $\mathcal{A}_{LA}$ computes the nested vector $\vec{N}_j = \{n_j^{\ell}\}_{\ell=1}^{L}$, where $n_j^1 = H_0(m_j^1)$ and $n_j^{\ell} = H_0(m_j^{\ell} \| H_0(n_j^{\ell-1}))\ ,\ \forall \ell = 2, \ldots, L$. $\mathcal{A}_{LA}$ computes the signatures $\sigma_{LA,j}^{1,L}$ and $\sigma_{PQ,j}$ by passing $(\vec{N}_j, t)$ and $(n_j^L \| s_{LA,j}^L, t)$ to $\texttt{LA−HASES.ASig}_{sk}$ and PQ−HASES.Sig, respectively.

If $\mathcal{A}$ wins the H−EU−CMA experiment, then it returns $q_s + 1$ valid signatures $\sigma^*_{HY,j} = (\sigma^{*\,1,L}_{LA,j}, \sigma^*_{PQ,j})$ on distinct batches $\vec{M}_j$ such that $\texttt{LA−HASES.AVer}(PK_{LA}, \vec{N}_j, \sigma^{*\,1,L}_{LA,j}) = 1$ and $\texttt{PQ−HASES}(PK_{PQ,j}, n_{j,L} \| s^{*\,L}_{LA,j}) = 1$. As a result, $\mathcal{A}_{LA}$ can extract q_S+1 valid signatures under LA−HASES on distinct messages. Thus, $Adv^{\text{H−EU−CMA}}_{\text{HY−HASES}}(t, q_s, q'_s) \leq Adv^{\text{A−EU−CMA}}_{\text{LA−HASES}}(t, q_s, q'_s)$. The same result applies to PQ−HASES: $Adv^{\text{H−EU−CMA}}_{\text{HY−HASES}}(t, q_s, q'_s) \leq Adv^{\text{F−EU−CMA}}_{\text{PQ−HASES}}(t, q_s, q'_s, 1)$. We conclude that:

$$Adv^{\text{H−EU−CMA}}_{\text{HY−HASES}}(t, q_s, q'_s) \leq \min\{Adv^{\text{A−EU−CMA}}_{\text{LA−HASES}}(t, q_s, q'_s)\ ,\ Adv^{\text{F−EU−CMA}}_{\text{PQ−HASES}}(t, q_s, q'_s, 1)\}$$

$Adv^{\text{H−EU−CMA}}_{\text{HY−HASES}}(t, q_s, q'_s) \leq \min\{J \cdot Adv^{\text{EU−CMA}}_{HORS}(t'_{PQ}, q_s.q'_s)\ ,\ Adv^{DL}_{G,\alpha}(t'_{LA})\}$, where $t'_{LA} = O(t) + O(\kappa^3)$ and $t'_{PQ} = O(t) + k \cdot H_0$. ■