Towards Practical Post-quantum Signatures for Resource-Limited Internet of Things

Rouzbeh Behnia
University of South Florida
Sarasota, Florida, USA
behnia@usf.edu

Attila A. Yavuz
University of South Florida
Tampa, Florida, USA
attilaayavuz@usf.edu

ABSTRACT

A digital signature is an essential cryptographic tool to offer authentication with public verifiability, non-repudiation, and scalability. However, digital signatures often rely on expensive operations that can be highly costly for low-end devices, typically seen in the Internet of Things and Systems (IoTs). These efficiency concerns especially deepen when post-quantum secure digital signatures are considered. Hence, it is of vital importance to devise post-quantum secure digital signatures that are designed with the needs of such constraint IoT systems in mind.

In this work, we propose a novel lightweight post-quantum digital signature that respects the processing, memory, and bandwidth limitations of resource-limited IoTs. Our new scheme, called ANT, efficiently transforms a one-time signature to a (polynomially-bounded) many-time signature via a distributed public key computation method. This new approach enables a resource-limited signer to compute signatures without any costly lattice operations (e.g., rejection samplings, matrix multiplications, etc.), and only with a low-memory footprint and compact signature sizes. We also developed a variant for ANT with forward-security, which is an extremely costly property to attain via the state-of-the-art post-quantum signatures.

KEYWORDS

Digital signatures; post-quantum security; authentication

ACM Reference Format:
Rouzbeh Behnia and Attila A. Yavuz. 2021. Towards Practical Post-quantum Signatures for Resource-Limited Internet of Things. In *Proceedings of ACM Conference (Conference'21).* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Efficient authentication and integrity are vital requirements to protect emerging IoT systems and cyber-critical infrastructures against common attacks such as man-in-the-middle, impersonation, data tampering, and many others. A reasonable measure to provide these properties is via message authentication codes. However, this symmetric primitive, while very efficient, requires pair-wise key distribution and storage, and fails to provide non-repudiation and public verifiability which are often required by many IoT applications like medical devices [1] and payment systems [2].

Digital signatures provide public verifiability and non-repudiation while being widely scalable, therefore an ideal solution to provide authentication and integrity for IoT applications. However, such schemes usually require expensive operations that can make the cryptographic overhead intolerable for some IoT applications, especially with those involving battery-powered and/or low-end devices (e.g., [1]).

Following Shor's algorithm [3], cryptosystems based on conventional hard problems such as elliptic curve discrete logarithm problem (e.g., ECDSA [4]) will be broken with the emergence of quantum computers. Therefore, NIST has started rounds of standardizations for post-quantum cryptography.[1] Hence, to ensure long-term security, resistance to quantum attacks should be considered. However, following NIST's third round of standardizations, the most efficient signature schemes currently in the competition (e.g., Dilithium [5]), could be very expensive for some IoT applications. For instance, for a resource-limited battery-powered medical sensor [6], that periodically generates and signs sensitive medical readings to be verified by a cloud service provider, the efficiency of the signing algorithm directly translates to a longer battery life.

Therefore, an ideal post-quantum secure signature for low-end IoT settings (e.g., battery-powered devices) should have the following desired properties: (i) High computation, memory, and bandwidth efficiency to minimize the burden of cryptography on the intended IoT application. This may, for instance, translate into a longer operation time for battery-powered devices due to the reduced energy consumption. (ii) It is not uncommon for low-end IoT devices to operate in an adversarial environment where they may be breached by an attacker via malware infiltration or physical means. To this end, providing compromise-resiliency features such as forward-security and side-channel resiliency are important.

The main goal of this paper is to create post-quantum signer-efficient and compact digital signature schemes with forward-security to meet the computation, memory, bandwidth, and battery needs of resource-limited IoT devices while minimizing the interventions and the cryptographic overhead.

1.1 Research Gap

In the classical domain (i.e., cryptosystems based on conventional hard problems), there have been many successful algorithmic and/or implementation attempts in proposing efficient signature schemes that are designed with signer efficiency in mind. However, as alluded to, in the case of post-quantum secure schemes, the existing

[1] https://csrc.nist.gov/projects/post-quantum-cryptography

Conference'21, June 2021, ,

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Table 1: Performance comparison of ANT and its lightweight counterparts on 8-bit Microcontroller

Scheme	Signing Cycles	Private Key (KB)	Signature (KB)	Post-Quantum Promise	Device	Rejection/Gaussian (Sampling)
ECDSA [7]	81, 324, 870	0.03	0.06	×	ATmega2560 (16 MHz)	×
Ed25519 [8]	33, 918, 780	0.03	0.06	×	ATmega2560 (16 MHz)	×
SchnorrQ [9]	5, 211, 321	0.03	0.06	×	ATmega2560 (16 MHz)	×
ESEM† [10]	616, 896	0.03	0.047	×	ATmega2560 (16 MHz)	×
BLISS-I [11]	10, 537, 981	0.25	0.7	✓	ATxmega128A1 (32 MHz)	✓
ANT-II	657, 517	0.03	0.432	✓	ATmega2560 (16 MHz)	×
ANT-FS-II	718, 678	0.09	0.432	✓	ATmega2560 (16 MHz)	×

† The current parameter sets of ESEM have be shown to be insecure. Further discussion is provided in Section 6.

post-quantum candidates (in the 3rd round of NIST post-quantum standardization process) (e.g., [5, 12, 13]), while being elegant, are not designed with resource-constrained IoT systems in mind. For instance, while the current hash-based standards/candidates have been extensively studied and improved both from the algorithmic and implementation perspectives [14–17], the slow signing and the large signature sizes may not be suitable for low-end devices operating on low-bandwidth networks. For instance, SPHINCS+ [14] signature size is around 17, 000 KB while its signature generation (on commodity hardware) is about two magnitudes of times slower than the lattice-based counterparts (e.g., Dilithium [5]). As another example, one can also consider the lattice-based candidates [5, 12] currently in the third round of standardization. They are more efficient than their hash-based counterparts for the signing speed and signature size, but they still require rather expensive operations (e.g., vector-matrix multiplications). While there have been attempts to enhance the performance, for instance, by improving the Number Theoretic Transform (NTT) operation (e.g., in [5]), when these schemes are implemented on low-end devices, they often incur high computation and communication overhead; which aside from the high end-to-end delay, directly translates to a lower operation time for battery-powered devices.

1.2 Our Contribution

We propose a new signer-efficient (lightweight) signature scheme with post-quantum security called ANT. ANT is specifically designed to provide computation and energy-efficient signing with compact signatures for resource-limited IoT devices. The main idea is to eliminate the public key generation, storage, certification, and transmission from the signer while keeping the efficiency of a near-optimal lattice-based one-time signature [18]. Another reason for choosing this particular one-time signature is that it is based on a very well-studied problem (e.g., shortest integer solution problem) that a current lattice-based candidate [5] is also based on. Specifically, we introduce a distributed public key construction algorithm, which exploits the key aggregation property of one-time lattice-based signature via a set of honest-but-curious servers, without requiring any intervention from the signer. This saves the signer from the burden of generating and transmitting one-time public keys, thereby offering a near-optimal yet polynomially-unbounded number signing capability. We also propose a forward secure variation of ANT that vastly enhances its breach-resiliency. Lastly, we propose a Merkle tree based variant to enable an efficient certification , which is often omitted in the context of one-time signatures.

1.3 Desirable Properties

Some of the desirable properties of our schemes are as follows:
(i) Near-optimal signing efficiency: The signature generation of ANT is almost as efficient as its underlying (near-optimal) one-time signature. This high signing efficiency directly translates to a lower energy consumption and longer operation time when implemented on battery-powered devices. As depicted in Table 1, ANT is 8× and 16× faster than its most efficient conventional (i.e., SchnorrQ [9]) and post-quantum (BLISS [11]) counterparts, respectively, when implemented on 8-bit microcontroller. The signing of ANT is also comparable to one of the most efficient conventional (EC-based) signatures with distributed verification called ESEM [10], and this is even without considering the recent attacks on the optimizations used in ESEM. When implemented on commodity hardware (see Table 2), ANT is at least 10× faster than its counterparts, currently in the third round of NIST PQC standardization.
(ii) Memory efficiency: ANT is memory efficient for the signer. Firstly, the private key size of ANT is as small as those in ECDSA and SchnorrQ [9]. Secondly, since ANT does not require the signer to compute a commitment (unlike done in BLISS [11] or Dilithium [5]), it has minimal (compared to its post-quantum counterparts) memory expansion at the signer's side. Moreover, unlike some online/offline and token-based schemes, ANT does not require the signer to generate and store tokens that will be depleted after a certain number of signatures are generated [19].
(iii) Compact signature: ANT and its variants, unlike its lattice-based counterparts (BLISS [11] or Dilithium [5]), are not based on the *Fiat-Shamir with aborts* (FSA) paradigm [20]. In the FSA paradigm, the masking term, which directly affects the signature size, needs to have a large norm to fully mask the private key. Therefore, ANT and its variants enjoy from a smaller signature size than their post-quantum counterparts. For instance, the signature size of ANT-II is about 5.5× smaller than the ones in Dilithium-II.
(iv) Side-channel resiliency: Current lattice-based signatures [20] (e.g., [5, 21]) based on the FSA paradigm rely on methods such as Gaussian or rejection sampling that are shown to be prone against side-channel attacks [22–24]. Moreover, there exist side-channel attacks exploiting the weak random number generators (specially in low-end devices) in signatures based on Fiat-Shamir transform

[25]. However, ANT does not require any Gaussian or rejection sampling during the signing. Moreover, the private key components are generated deterministically, and therefore, it is not prone to the common side-channel attacks mentioned above.

(v) Forward Security: The forward security property ensures the authenticity and integrity of the data items before an attack point. It is achieved by evolving the secret key periodically (e.g., every hour or per signing). Forward secure signatures can substantially enhance the breach resiliency of critical infrastructures, and therefore have various applications in IoTs and forensics (e.g., [26]). *To our knowledge,* ANT-FS *is the first signer-optimal polynomially-bounded lattice-based signature with forward security.*

(v) Improved Post-quantum Security: ANT, unlike lattice-based schemes [5, 21] relying on the FSA paradigm [20], does not use random oracles model (ROM). As it is shown in [27], the transformation to provide quantum security proof for signature schemes in the ROM could incur some efficiency loss. Since ANT security is not based on such assumption, the post-quantum security is achieved without a significant efficiency loss.

1.4 Limitations

The high signer efficiency in ANT is achieved by moving the burden of public key computation to a set of servers. More specifically, the verifier needs to contact the server(s) to obtain the public key of the signer corresponding to a particular signature. The high bandwidth requirement for the servers and the verifier is due to the fact that the public key communication overhead in the new scheme is per signature. This should be particularly considered for communication-heavy applications. However, as stated in our use-cases, we assume the verifier's machine is at least a commodity hardware with a stable high-bandwidth connection. Note that in our scheme, the signer does not need to interact with any party to generate the signature. We assume that servers are honest-but-curious and t-private (Definition 4). That is, ANT will remain secure even if $(t-1)$ parties collude. ANT is a suitable candidate for applications where highly efficient signing is a priority and few milliseconds of delay on the verifier's side (due to the interaction with servers) is tolerable.

2 PRELIMINARIES

Notation. We work on a ring $\mathcal{R} = \mathbb{Z}_q[x]/(x^n+1)$ for a prime q with n being a power-of-two. Each element of $\mathcal{R}$ is represented as a polynomial of degree $n-1$. The scheme is parametrized by $n, q, k, l, \beta_{\mathsf{xy}}, \beta_{\mathsf{c}}$ and β_{ver}. We denote vectors as bold letters (i.e., $\mathbf{a}$) while scalars are denoted non-bold. $a \xleftarrow{\$} \mathcal{D}$ denotes a is being sampled randomly from set the $\mathcal{D}$. $|a|$ denotes the bit length of a, i.e., $|a| = \log_2 a$. For a vector $\mathbf{a} = (a_1, \ldots, a_n)$ we define $\|\mathbf{a}\|_\infty = max\{|a_i| : i = 1, \ldots, n\}$. We use κ to denote our security parameter.

We define $\mathcal{K} \subset \mathcal{R}$ with small coefficients (i.e., $\leq \beta_{\mathsf{xy}}$) as our private key space. $\mathcal{S} \subset \mathcal{R}$ is defined as our signature space with coefficients $\leq \beta_{\mathsf{ver}}$. We use $\mathsf{H}_1(\cdot)$ to map arbitrary length messages to our message space $\mathcal{M}$ with vectors with exactly β_{c} number of 1 or -1 with the rest fo the elements being 0. We also set $\mathsf{H}_2 : \{0,1\}^* \times \mathbb{Z}_p \rightarrow \{0,1\}^*$. We define a Pseudo Random Function $\mathsf{PRF}_1 : \{0,1\}^* \rightarrow \{0,1\}^*$.

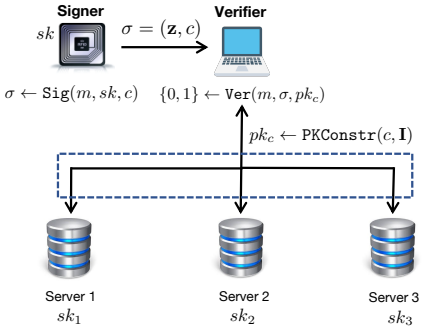

Figure 1: System model with three public key servers

Definition 1. Given the ring $\mathcal{R}$ and a matrix $\mathbf{A} \in \mathcal{R}^{k \times l}$, the Small Integer Solution over Rings problem (Ring-SIS$_{q,n,k,l,\beta_{\mathsf{xy}}}$) asks to find a non-zero vector $\mathbf{s} \in \mathcal{R}^l$ such that $\|\mathbf{s}\|_\infty \leq \beta_{\mathsf{xy}}$ and $\mathbf{As} = 0 \bmod q$.

Definition 2. A signature scheme consists of four algorithms SGN = (Setup,Kg,Sig,Ver) defined as follows.

- $params \leftarrow$ SGN.Setup(1^κ): Given the security parameter 1^κ, it sets up the systems by outputting the system parameters $params$.
- $(sk, pk) \leftarrow$ SGN.Kg($params$): Given $params$, it outputs the private and public key pair (sk, pk).
- $\sigma \leftarrow$ SGN.Sig(m, sk): Given the message m and the signer's private key sk, it outputs the signature σ.
- $\{0,1\} \leftarrow$ SGN.Ver(m, σ, pk): Given a message-signature pair (m, σ), and the claimed signer's public key pk, it outputs a decision bit $\{0,1\}$.

We define Lyubashevsky and Micciancio one-time signature (LM-OTS) scheme [18] as follows.

Definition 3. *Lyubashevsky and Micciancio one-time signature* LM-OTS = (Setup, Kg,Sig,Ver) *is defined as follows.*

- $params \leftarrow$ LM-OTS.Setup(1^κ): Given security parameter κ, system setup starts by selecting the parameters q, n, k, l, and the ring $\mathcal{R}$. It then selects $\mathbf{A} \in \mathcal{R}^{k \times l}$ and publishes $params = (q, n, k, l, \mathbf{A})$ for all the users in the systems.
- $(sk, pk) \leftarrow$ LM-OTS.Kg($params$): The user selects $[\mathbf{s}_1, \mathbf{s}_2] \in \mathcal{K}^{l \times 2}$ as their private key and compute $\mathbf{t} = \mathbf{As}_1 \in R$, $\mathbf{t}' = \mathbf{As}_2 \in R$. It sets $sk \leftarrow [\mathbf{s}_1, \mathbf{s}_2]$ and $pk \leftarrow (\mathbf{t}, \mathbf{t}')$.
- $\mathbf{z} \leftarrow$ LM-OTS.Sig(sk, m): Given sk and m, the signer first computes the signature as $\mathbf{z} \leftarrow \mathbf{s}_1 \mathsf{H}_1(m) + \mathbf{s}_2$.
- $\{0,1\} \leftarrow$ LM-OTS.Ver($m, \mathbf{z}, pk$): Given a message-signature pair $(m, \mathbf{z})$ and the public key pk, the verifier checks if $\mathbf{A}^T\mathbf{z} = \mathbf{t}\mathsf{H}_1(m) + \mathbf{t}'$ holds, outputs 1, else outputs 0.

Definition 4. *A protocol is called computational t-private [28] if it is computationally hard for any subset of servers Γ where $|\Gamma| \leq t$, to produce/compute any information other than what could have been computed individually from their set of private inputs.*

3 MODELS

In this section, we define our system and security models.

3.1 System Model

Our system model is designed around heterogeneous IoT systems where battery-powered and low-end devices, which may need to operate for long periods of time, broadcast signed measurements to the resourceful verifiers (e.g., a laptop). As shown in Fig. 1, there are three type of entities in our system model: (i) The singer which is deemed to be highly resource-limited. *The signer is not expected to store and/or communicate the public keys*, it only uses its private key to compute the signature σ_c, for period c. This is crucial to enable efficient signing and meet the stringent requirements of the low-end signer in our system model. (ii) The verifier is considered to be a rather resourceful machine (e.g., commodity hardware) with high bandwidth connection, capable of communicating with the public key servers. (iii) Lastly, t distinct *honest-but-curious* ($t = 3$ in Fig. 1) and t-private servers (Definition 4).

After the initialization step, we assume the existence of a network that consists of a verifier and our servers. In our model, the public key pk_c is constructed in a distributed fashion. The verifier can query for public key components $pk_{j,c}$ (for $1 \leq j \leq t$) from the distributed servers before or after receiving the signature σ_c from the signer.

We call our model a signature scheme with *distributed public key computation* (DPC). In addition to the algorithms defined in Definition 2, the new scheme has a public key construction algorithm $\texttt{PKConstr}(\cdot)$ that enables the verifier to obtain the public key for a particular signature. We will discuss different variations of our models with different features/properties in Section 4.

Definition 5. A signature scheme with distributed public key computation consists of four algorithms $\texttt{SGN-DPC} = (\texttt{Kg}, \texttt{Sig}, \texttt{Ver}, \texttt{PKConstr})$ defined as follows.

- $params \leftarrow \texttt{SGN-DPC.Setup}(1^\kappa)$: Given the security parameter 1^κ, it sets up the systems by outputting the system parameters $params$.
- $(sk, c.\langle sk_1, \ldots, sk_t\rangle) \leftarrow \texttt{SGN-DPC.Kg}(params)$: Given the parameter $params$, the signer computes the initial state c and its private key sk. Using the private key sk it computes the t private key components $\{sk_1, \ldots, sk_t\}$ to be securely transmitted to the servers. Note that this algorithm would only takes place once to sign polynomial number of signatures.
- $\sigma \leftarrow \texttt{SGN-DPC.Sig}(m, sk, c')$: Given a message m, the signer's private key sk and the current state c', it outputs the signature $\sigma = (\mathbf{z}, c)$, where c is the updated state.
- $pk_c \leftarrow \texttt{SGN-DPC.PKConstr}(c, \mathbf{I})$: Given c and server indexes $\mathbf{I} = \langle 1, \ldots, t\rangle$, it queries the partial public key $pk_{j,c}$ that corresponds to the c^{th} signature, for each server in $\mathbf{I}$ and computes the final public key pk_c for state c.
- $\{0, 1\} \leftarrow \texttt{SGN-DPC.Ver}(m, \sigma, pk_c)$: Given a message-signature pair (m, σ), and the claimed signer's public key pk_c, obtained from the above algorithm, it outputs a decision bit $\{0, 1\}$.

3.2 Security Model

Following the system model given above, we assume our t public key servers are t-private (Definition 4). We also assume that our servers are honest-but-curious where they follow the protocol but strive to learn as much as possible from the information being received, observed or shared.

In the following definition, we define the security of signature schemes. Our only deviation of the standard EU-CMA definition [29] is that we equip $\mathcal{A}$ with two additional oracles defined below.

- $\texttt{CrptServer}(j)$: For $1 \leq j \leq t$, this oracle provides $\mathcal{A}$ with the secret seed associated with server j. This oracle can be queried for up to $t-1$ different servers and it is essential to capture the t-private property of our DPC model.
- $\texttt{PKConstr}(st, j)$: Given the state c, and $1 \leq j \leq t$, this oracle returns the partial public key $pk_{j,c}$.

After the initialization phase (i.e., $\texttt{SGN-DPC.Setup}(\cdot)$ and $\texttt{SGN-DPC.Kg}(\cdot)$), $\mathcal{A}$ is given access to the signature generation oracle, $\texttt{CrptServer}(\cdot)$ and $\texttt{PKConstr}(\cdot)$.

$\mathcal{A}$ wins, if it outputs a *valid* message-signature pair (that was not previously outputted from the sign oracle) after making polynomially-bounded number of queries.

Definition 6. Existential Unforgeability under Chosen Message Attack (EU-CMA) experiment for a signature scheme with distributed verification $Expt^{EU-CMA}_{\texttt{SGN-DPC}}$ is defined as follows.

- $params \leftarrow \texttt{SGN-DPC.Setup}(1^\kappa)$
- $(sk, st) \leftarrow \texttt{SGN-DPC.Kg}(params)$
- $(m^*, \sigma^*) \leftarrow \mathcal{A}^{\texttt{SGN-DPC.Sig}(\cdot), \texttt{CrptServer}(\cdot), \texttt{PKConstr}(\cdot)}(\cdot)$
- If $1 \leftarrow \texttt{SGN-DPC.Ver}(m^*, \sigma^*, pk)$, m^* was not queried to $\texttt{SGN-DPC.Sig}(\cdot)$, and $\texttt{CrptServer}(\cdot)$ has been only called on maximum of $t-1$ servers, return 1, else, return 0.

The EU-CMA advantage of $\mathcal{A}$ is defined as $Adv^{EU\text{-}CMA}_{\texttt{SGN-DPC.}} = \Pr[Expt^{EU-CMA}_{\texttt{SGN-DPC}} = 1]$.

Forward Security. A forward secure (FS) signature scheme is a key-evolving digital signature scheme where the operation of the signature generation is divided into time periods, and each time period uses a different private key to sign a message [30]. Hence, FS signature scheme has an additional key update algorithm to evolve the secret key at the end of each time period. In our model, since we update the secret key after each instance of signature generation, to reduce the complexity of our model, we incorporate the private key update functionality in both $\texttt{SGN-DPC.Sig}(\cdot)$ and $\texttt{SGN-DPC.PKConstr}(\cdot)$. Where in $\texttt{SGN-DPC.Sig}(\cdot)$ the update happens after every instance of signature generation algorithm and in $\texttt{SGN-DPC.PKConstr}(\cdot)$, happens after each public key query on distributed servers.

The security model of FS signatures is similar to the model in Definition 6 except that the adversary is equipped with an additional oracle, $\texttt{BreakIn}(\cdot)$, which returns the current private key to the adversary $\mathcal{A}$. In the case of our scheme, if the adversary has queried j signature queries to the signing oracle, the $\texttt{BreakIn}(\cdot)$ oracle will return the $(j+1)$-th private key to the adversary. In addition to the winning condition in Definition 6, the secret key corresponding to the forgery signature should have never been outputted by $\texttt{BreakIn}(\cdot)$ (i.e., to prevent trivial forgery).

4 PROPOSED SCHEME

The main requirement for our proposed scheme is to ensure communication, computation and storage efficiency for the signer, while maintaining a reasonable overhead for the verifiers with a distributed public key computation. We assume that the verifiers are resourceful, as it is typically the case for many IoT applications

(e.g., resourceful servers collect the data and verify it). Our design objectives are as follows: *(i)* An efficient signing algorithm that avoids costly operations (e.g., rejection sampling) and only requires efficient symmetric key based operations and efficient arithmetics. *(ii)* Avoiding online/offline methods that require the signer's intervention after a certain time period (e.g., singing T signatures). *(iii)* The signer should not store precomputed tokens (e.g., unlike online/offline signatures, optimization tables) for efficient memory usage. *(iv)* The signature should be small-constant size.

We achieve the above properties by efficiently transforming an efficient one-time signature [18] to a polynomially-bounded many time signature. This is done by shifting the public key generation and communication to a set of t public key construction servers where the verifier would communicate with before attempting to verify a signature. We emphasis that, after offline initialization phase, the signer never interacts with the public key servers, and operates as in traditional signature schemes. This is a key design property for ANT to achieve near-optimal signer efficiency.

4.1 The Basic Scheme

In this section, we describe our basic scheme ANT, followed by its forward security version ANT-FS and variations to allow for more efficient public key certification methods. and $\mathsf{Samp}_1 : \{0,1\}^* \rightarrow \mathcal{K}'^l$ and $\mathsf{Samp}_2 : \{0,1\}^* \rightarrow \mathcal{K}'^l$. The distribution of $\mathcal{K}'$ is computed based on the number of public key servers and $\mathcal{K}$.

We propose a distributed post-quantum signature scheme with efficient signature generation for low-end devices. We present our basic scheme ANT in Algorithm 1. As depicted in Algorithm 1, in the $\mathsf{ANT.Kg}(\cdot)$ algorithm, the signer will generate t private key components $(sk_1, \ldots, sk_t)$ and sends them to the t servers. Note that the servers could also be preloaded with these secret keys, upon the production of the low-end device.

The signature generation of the one-time scheme [18], presented in Definition 3, requires the signer to generate a new public key for each signature. This significantly increases signer computation and transmission overhead (signature plus public key per message), and likely inhibits its practice uses for low-end IoT systems. We address this significant limitation by leveraging the additive homomorphism of the keys in [18] to shift the public key generation burden to the distributed servers in our model. Therefore, the signature generation algorithm $\mathsf{ANT.Sig}(\cdot)$ is highly efficient and does not require any involvement of the server and is completely non-interactive. The verifier can initiate the $\mathsf{ANT.PKConstr}(\cdot)$ algorithm to obtain the public key pk_c for the time period c by interacting with the t servers. Note that the verifier can initiate this algorithm at any time (e.g., before the time period c).

After obtaining the public key pk_c, the verifier initiates $\mathsf{ANT.Ver}(\cdot)$, which is again non-interactive, to verify the signature.

4.2 The Forward Security Scheme

Forward secure signatures are designed to improve the breach resiliency of cyber infrastructures by mitigating the consequences of attacks where private keys (signing keys) are compromised. The main idea is therefore to update the private key(s) after each signature generation/public key query such that the new key cannot be used to sign for past time periods.

Algorithm 1 ANT: Basic Scheme

$(params) \leftarrow \mathsf{ANT.Setup}(1^\kappa)$: Given 1^κ, generate the parameters based on $\mathsf{LM\text{-}OTS.Setup}(1^\kappa)$.

return $params = (q, n, k, l, \beta_{\mathsf{xy}}, \beta_{\mathsf{ver}}, \beta_{\mathsf{c}}, \mathbf{A} \in \mathcal{R}_q^{k \times l})$

$(sk, c.\langle sk_1, \ldots, sk_t \rangle) \leftarrow \mathsf{ANT.Kg}(params)$: Given 1^κ, generate the signer's private key and the private seeds for the t servers.

$sk \xleftarrow{\$} \{0,1\}^\kappa$
Set state $c = 0$
for $j \in \mathbf{I}$ where $\mathbf{I} = \{1, \ldots, t\}$ **do**
    $sk_j \leftarrow \mathsf{PRF}_1(sk, j)$
Send the private key components $\{sk_1, \ldots sk_t\}$ to the t servers
return (sk, c)

$\sigma \leftarrow \mathsf{ANT.Sig}(m, sk, c)$: Run by the singer to issue a signature on m.

$c \leftarrow c + 1$
for $j = 1, \ldots, t$ **do**
    $sk_j \leftarrow \mathsf{PRF}_1(sk, j)$
    $\mathbf{x}_{j,c} \leftarrow \mathsf{Samp}_1(sk_j, c), \mathbf{y}_{j,c} \leftarrow \mathsf{Samp}_2(sk_j, c)$
    $\mathbf{x}_c \leftarrow \mathbf{x}_c + \mathbf{x}_{j,c}, \mathbf{y}_c \leftarrow \mathbf{y}_c + \mathbf{y}_{j,c}$
$\mathbf{z} \leftarrow \mathbf{x}_c \mathsf{H}_1(m, c) + \mathbf{y}_c$
return $\sigma = (c, \mathbf{z})$

$pk_c \leftarrow \mathsf{ANT.PKConstr}(c, \mathbf{I})$: This algorithm is initiated by the verifier. Given c and $j \in \mathbf{I}$ each server returns $(\mathbf{t}_{c,j}, \mathbf{t}'_{c,j})$ and the verifier computes the full public key pk_c.

At server j **do**:
    $\mathbf{x}_{j,c} \leftarrow \mathsf{Samp}_1(sk_j, c), \mathbf{y}_{j,c} \leftarrow \mathsf{Samp}_2(sk_j, c)$
    $\mathbf{t}_{j,c} \leftarrow \mathbf{x}_{j,c}\mathbf{A}, \mathbf{t}'_{j,c} \leftarrow \mathbf{y}_{j,c}\mathbf{A}$
    Send $pk_{j,c} \leftarrow (\mathbf{t}_{j,c}, \mathbf{t}'_{j,c})$ to the verifier
Verifier computes $\mathbf{t}_c \leftarrow \mathbf{t}_c + \mathbf{t}_{j,c}, \mathbf{t}'_c \leftarrow \mathbf{t}'_c + \mathbf{t}'_{j,c}$
return $pk_c = (\mathbf{t}_c, \mathbf{t}'_c)$

$\{0,1\} \leftarrow \mathsf{ANT.Ver}(m, \sigma, pk_c)$: Given pk_c obtained from the above algorithm, the verifier verifies the signature as follows.

if $\|\mathbf{z}\|_\infty \leq \beta_{\mathsf{ver}}$ **then**:
    Parse pk_c as $(\mathbf{t}_c, \mathbf{t}'_c)$
    if $\mathbf{Az} = \mathbf{t}_c \mathsf{H}_1(m, c) + \mathbf{t}'_c$ holds, **then return** 1
else return 0

In this section, we present a forward secure variation of ANT, called ANT-FS, in Algorithm 2. The initial seed is generated as a κ bit string sk. sk is then used to generate server seeds $sk_{j,c}$ (Step 4 in $\mathsf{ANT\text{-}FS.Kg}(\cdot)$) and then deleted immediately (Step 5 in $\mathsf{ANT\text{-}FS.Kg}(\cdot)$). ANT-FS achieves forward security by using a hash chain on the server's seeds on the signer's side during signature generation and evolving the seeds on the server's sides using a new public key construction algorithm $\mathsf{ANT\text{-}FS.PKConstr}(\cdot)$. Additionally, ANT-FS enjoys from a fast singing since it does not require the regenerating of the server seeds via PRF calls. However, this comes with the cost of having a linear (to the number of servers) private key size. Note that the verification algorithm for ANT-FS is identical as in the original scheme in Algorithm 1.

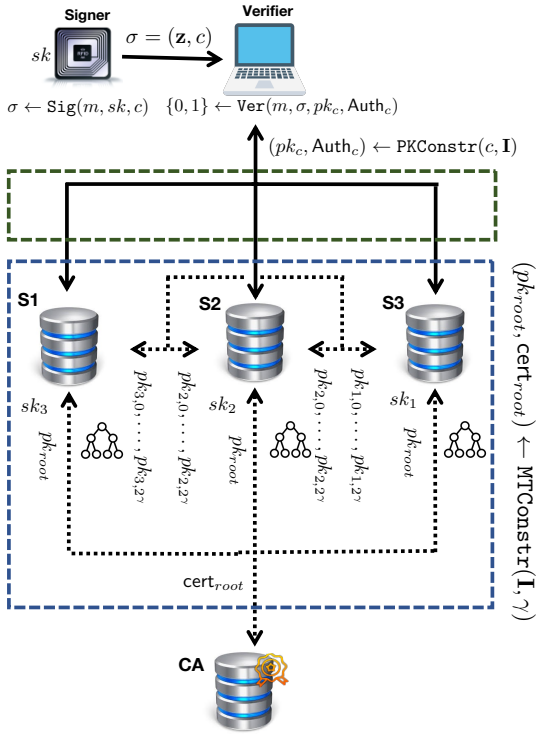

Figure 2: Batch Certification with Merkle Tree

4.3 Signer independent offline certification management

As discussed, the scheme proposed in Section 4 is the base version of ANT, where the verifier contacts each server through the ANT.PKConstr() algorithm to receive the corresponding public key pk_c . We do not consider public key certification in the base version of the scheme proposed in Algorithm 1. In this section, we provide two variations of ANT which focus on different public key certification methods.

There has been a wide array works to efficiently transform one-time signatures to multiple-time [31] or polynomially-bounded many time signatures [32, 33]. However, generally, whenever a one-time signature is used arbitrary number of times, it will require arbitrary number of public keys and certificates. This problem is addressed in hash-based signatures by bounding total number of signatures (albeit polynomially-bounded in [32]) and building a Merkle tree MT on top. This is the main reason that such schemes suffer from very large signatures. Here, due to the main purpose of the scheme to be suitable of resource-limited devices (often operating on low-bandwidth networks), we avoid very large signatures by presenting two methods that enable public key certification without the involvement or incurring additional costs on the (low-end) signer in our design.

4.3.1 Individual Public Key Certification. In order to provide public key certification, we propose a variation of the ANT.PKConstr($\cdot$) in Algorithm 4. In the new ANT.PKConstr($\cdot$) algorithm, every server $j \in \mathbf{I}$ computes their public key components $pk_{j,c}$ using the provided secret seed and sends it to all other servers. In the second step, all other servers would use these partial public key components to compute the final public key pk_c and send it to the certificate authority (CA) to issue the certificate cert_c. While this method has optimal communication overhead, it requires interaction with the CA (by the distributed public key servers) for a verification request.

The unique design of the new scheme enables the utilization of a precomputation method to improve the performance of the verification algorithm in both ANT and ANT-FS. More specifically, given the signer in our schemes is not involved in the verification algorithm and since ANT.PKConstr($\cdot$) does not depend on the message, in the certification methods in Algorithm 4, the servers can precompute pk_c for the future 2^γ signatures (i.e., $c = 1, \ldots, 2^\gamma$). The public keys pk_c along with their corresponding certificate cert_c can then

Algorithm 2 ANT-FS: The Forward Secure Variant

$(params) \leftarrow \mathtt{ANT\text{-}FS.Setup}(1^\kappa)$: This algorithm is identical to the one in Algorithm 1.

$(c, \langle sk_{1,c}, \ldots sk_{t,c} \rangle) \leftarrow \mathtt{ANT\text{-}FS.Kg}(params)$: Given 1^κ, generate the signer's private key and the private seeds for the t servers.

$sk \xleftarrow{\$} \{0,1\}^\kappa$
Set state $c = 0$
for $j \in \mathbf{I}$ where $\mathbf{I} = \{1, \ldots, t\}$ **do**
  $sk_{j,c} \leftarrow \mathsf{PRF}_1(sk, j)$
delete sk
Send private key components for period c, $\{sk_{1,c}, \ldots sk_{t,c}\}$, to the t servers.
return $sk \leftarrow \{sk_{1,c}, \ldots sk_{t,c}\}$ and state c

$\sigma \leftarrow \mathtt{ANT\text{-}FS.Sig}(m, sk)$: Run by the singer to issue a signature on m.

$c \leftarrow c + 1$
for $j = 1, \ldots, t$ **do**
  $\mathbf{x}_{c,j} \leftarrow \mathsf{Samp}_1(sk_{j,c}, c), \mathbf{y}_{c,j} \leftarrow \mathsf{Samp}_2(sk_{j,c}, c)$
  $\mathbf{x}_c \leftarrow \mathbf{x}_c + \mathbf{x}_{c,j}, \mathbf{y}_c \leftarrow \mathbf{y}_c + \mathbf{y}_{c,j}$
  $sk_{j,c+1} \leftarrow \mathsf{H}_2(sk_{j,c})$, **delete** $sk_{j,c}$
$\mathbf{z} \leftarrow \mathbf{x}_c \mathsf{H}_1(m, c) + \mathbf{y}_c$
return $\sigma = (c, \mathbf{z})$

$pk_c \leftarrow \mathtt{ANT\text{-}FS.PKConstr}(c, \mathbf{I})$: This algorithm is initiated by the verifier. Given c and $j \in \mathbf{I}$ each server returns $(\mathbf{t}_{c,j}, \mathbf{t}'_{c,j})$, updates its private key component, and the verifier computes the full public key pk_c.

At server j **do**:
  $\mathbf{x}_{j,c} \leftarrow \mathsf{Samp}_1(sk_{j,c}, c), \mathbf{y}_{j,c} \leftarrow \mathsf{Samp}_2(sk_{j,c}, c)$
  $\mathbf{t}_{j,c} \leftarrow \mathbf{x}_{j,c}\mathbf{A}, \mathbf{t}'_{j,c} \leftarrow \mathbf{y}_{j,c}\mathbf{A}$
  $sk_{j,c+1} \leftarrow \mathsf{H}_2(sk_{j,c})$
  Send $pk_{j,c} \leftarrow (\mathbf{t}_{j,c}, \mathbf{t}'_{j,c})$ to the verifier
Verifier computes $\mathbf{t}_c \leftarrow \mathbf{t}_c + \mathbf{t}_{j,c}, \mathbf{t}'_c \leftarrow \mathbf{t}'_c + \mathbf{t}'_{j,c}$
return $pk_c = (\mathbf{t}_c, \mathbf{t}'_c)$

$\{0,1\} \leftarrow \mathtt{ANT\text{-}FS.Ver}(m, \sigma, pk_c)$: This algorithm is identical to the one in Algorithm 1.

Algorithm 3 MT-ANT: Merkle Tree Based Scheme with Certification

$(params) \leftarrow$ MT-ANT.Setup(1^κ): This algorithm is identical to the one in Algorithm 1 except, in addition to all the parameters a parameter γ is also selected.

$(sk, c) \leftarrow$ MT-ANT.Kg$(params)$: This algorithm is identical to the one in Algorithm 1.

$(pk_{root}, \text{cert}_{root}) \leftarrow$ MT-ANT.MTConstr$(\mathbf{I}, \gamma)$:

for $j = 1, \ldots, t$ **do**
  for $c = 0, \ldots, 2^\gamma$ **do**
    $\mathbf{x}_{j,c} \leftarrow \text{Samp}_1(z_j, c), \mathbf{y}_{j,c} \leftarrow \text{Samp}_2(z_j, c)$
    $\mathbf{t}_{j,c} \leftarrow \mathbf{x}_{j,c}\mathbf{A}, \mathbf{t}'_{j,c} \leftarrow \mathbf{y}_{j,c}\mathbf{A}$
    $pk_{j,c} \leftarrow (\mathbf{t}_{j,c}, \mathbf{t}'_{j,c})$
    Send $pk_{j,c}$ to other $t-1$ servers
for $c = 0, \ldots, 2^\gamma$ **do**
  After receiving all $pk_{j,c}$, each server computes:
  $\mathbf{t}_c = \sum_{j=0}^{j=t} \mathbf{t}_{j,c}$ and $\mathbf{t}'_c = \sum_{j=0}^{j=t} \mathbf{t}'_{j,c}$
  $pk_c \leftarrow (\mathbf{t}_c, \mathbf{t}'_c)$
$(\mathsf{MT}, pk_{root}) \leftarrow$ MT.Init$(pk_c, \ldots, pk_{2^\gamma})$
All servers store the MTwith root pk_{root}
Send pk_{root} to the CA to obtain cert_{root}
We assume all users are initialized by pk_{root} and cert_{root}
return $(pk_{root}, \text{cert}_{root})$

$\sigma \leftarrow$ MT-ANT.Sig(m, sk, c): Run by the singer to issue a signature on m.

$c \leftarrow c + 1$
for $j = 1, \ldots, t$ **do**
  $sk_j \leftarrow \text{PRF}_1(sk, j)$
  $\mathbf{x}_{j,c} \leftarrow \text{Samp}_1(sk_j, c), \mathbf{y}_{j,c} \leftarrow \text{Samp}_2(sk_j, c)$
  $\mathbf{x}_c \leftarrow \mathbf{x}_c + \mathbf{x}_{j,c}, \mathbf{y}_c \leftarrow \mathbf{y}_c + \mathbf{y}_{j,c}$
$\mathbf{z} \leftarrow \mathbf{x}_c \mathsf{H}_1(m, c) + \mathbf{y}_c$
return $\sigma = (c, \mathbf{z})$

$(pk_c, \text{Auth}_c) \leftarrow$ MT-ANT.PKConstr$(c, \mathbf{I})$: This algorithm is initiated by the verifier. Given c , the servers return the leaf pk_c and the corresponding authentication path.

A server $j \in \mathbf{I}$ works as follows:
  Runs $(pk_c, \text{Auth}_c) \leftarrow$ MT.Output(c)
return (pk_c, Auth_c)

$\{0, 1\} \leftarrow$ MT-ANT.Ver$(m, \sigma, pk_c, \text{Auth}_c)$: Given pk_c obtained from the above algorithm, the verifier verifies the signature as follows.

if $1 \leftarrow$ MT.Verify$(pk_{root}, pk_c, \text{Auth}_c)$ **then**:
if $\|\mathbf{z}\|_\infty \leq \beta_{\text{ver}}$ **then**:
  Parse pk_c as $(\mathbf{t}_c, \mathbf{t}'_c)$
  if $\mathbf{Az} = \mathbf{t}_c \mathsf{H}_1(m, c) + \mathbf{t}'_c$ holds, **then return** 1
else return 0

be either stored on the servers to be downloaded on demand or directly downloaded to the verifiers'. This will significantly reduce the online overhead of ANT.PKConstr$(\cdot)$ algorithm.

Algorithm 4 Public Key Construction with Certification

$(pk_c, \text{cert}_c) \leftarrow$ ANT.PKConstr(c):

for $j = 1, \ldots, t$ **do**
  $\mathbf{x}_{j,c} \leftarrow \text{Samp}_1(sk_j, c), \mathbf{y}_{j,c} \leftarrow \text{Samp}_2(sk_j, c)$
  $\mathbf{t}_{j,c} \leftarrow \mathbf{x}_{j,c}\mathbf{A}, \mathbf{t}'_{j,c} \leftarrow \mathbf{y}_{j,c}\mathbf{A}$
  $pk_{j,c} \leftarrow (\mathbf{t}_{j,c}, \mathbf{t}'_{j,c})$
  Send $pk_{j,c}$ to other $t-1$ servers
After receiving all other $pk_{j,c}$ from severs compute: $\mathbf{t}_c \leftarrow \mathbf{t}_c + \mathbf{t}_{i,c}, \mathbf{t}'_c \leftarrow \mathbf{t}'_c + \mathbf{t}'_{i,c}$
$pk_c \leftarrow (\mathbf{t}_c, \mathbf{t}'_c)$
Send pk_c to the CA to receive cert_c
return (pk_c, cert_c)

4.3.2 Batch Certification with Merkle Tree. To minimize certificate request, our construction also allows for the adoption of Merkle Tree (MT) [34]. This well-studied method has been adopted in a number of other schemes as well (e.g, [33]). Generally, an MT-based scheme consists of the following algorithms:

- $(\mathsf{MT}, pk_{root}) \leftarrow$ MT.Init$(x_1, \ldots, x_n)$: Given n values $x_1, \ldots, x_n$, this algorithm will construct a tree of hight $|n|$ with n leaves and outputs MT and its root pk_{root}.
- $(pk_c, \text{Auth}_c) \leftarrow$ MT.Outputt(c): Given the index c, this algorithm outputs the leaf at location c, i.e., pk_c and its corresponding authentication path Auth_c.
- $\{0, 1\} \leftarrow$ MT.Verify$(, pk_c, pk_{root}, \text{Auth}_c)$: Given the root pk_{root}, a leaf at location c, i.e., pk_c, and its alleged authentication path, this algorithm outputs a decision bit $\{0, 1\}$.

We present our signature scheme with a Merkle tree called MT-ANT in Algorithm 3. The new scheme requires an additional algorithm called MT-ANT.MTConstr$(\cdot)$ that is run once (for every 2^γ signatures) among the servers to construct the MT by initiating the MT.Init$(\cdot)$ algorithm. As also depicted in Fig. 2, the servers collaborate to compute a MT by computing 2^γ public keys and then use them to construct the tree. For certification purposes, root can then be sent to the CA for certification. This method simplifies the public key certification process by requiring to contact the CA only once (to authenticate the root of the tree) to sign 2^γ messages. All the dashed lines in Fig. 2 are only executed once for every 2^γ messages signed. We note that once all the leaves are depleted, the servers, without the intervention of the signer, can generate a new tree.

Using this method, signature verification will require the servers to send the authenticating path along with the final public key pk_c, and therefore it incurs additional γ communication overhead. Note that, unlike other solutions, this does not imply that our scheme is 2^γ time since, after the depletion of 2^γ public keys, a new tree (of arbitrary size) can be generated by the server, without the interaction of the signer. One can see the direct relationship between γ (i.e., the size of the tree) and the communication overhead between the verifier and the public key generation servers. However, since the tree generation does not require any signer intervention, if the communication bandwidth is a concern, to improve verifier communication, once can keep γ smaller.

5 SECURITY ANALYSIS

Theorem 1. *If there exist an adversary $\mathcal{A}$ that breaks the EU-CMA property (as defined in Definition 6) of* ANT *in Algorithm 1, then one can construct another algorithm $\mathcal{B}$ that runs $\mathcal{A}$ as a subroutine and breaks the SIS problem.*

Proof. Our proof technique follows a similar approach in [18]. As state in Section 3, we assume the public key servers are computational t-private. We assume that the reduction algorithm $\mathcal{B}$ has knowledge of the private key components of the t servers (i.e., $z_1, \ldots, z_t$) secret. Therefore, given the knowledge of $\mathcal{B}$ of the private key components, one can see that it can simulate $\texttt{SGN-DPC.Sig}(\cdot)$ and $PKConstr(\cdot)$ oracles as in EU-CMA experiment (Definition 6) exactly as in the original scheme. $\mathcal{B}$ responds to $\texttt{CrptServer}(i)$ for $i \in \{1, \ldots, t\}$ queries by returning the corresponding private key components z_i.

Note that following Definition 4, $\mathcal{A}$ can only initiate $\texttt{CrptServer}(\cdot)$ queries on $t-1$ distinct servers. Consequently, if we assume $\mathcal{A}$ has knowledge on all but one server, then to guess the private key components it either has to guess the correct z_i or $(\mathbf{x}_i, \mathbf{y}_i)$. It can come up with a correct guess with probabilities $2^{-\kappa}$ and $2^{-2ln|\beta_{\mathsf{xy}}|}$. Therefore, without knowing at least one of the private key components pairs, the adversary has to correctly compute a valid partial signature $\mathbf{x}_i\mathsf{H}_1(m,c) + \mathbf{y}_i$. The scheme proposed in Algorithm 1 is based on the one-time signature obtained from a Ring-SIS based one-time signature similar to that in [18]. We therefore rely on the results of Theorem 3.2 in [18].

The theorem states that a OTS scheme is correct and secure if it meets the following properties.

- *Closure:* $\{\mathbf{z} \leftarrow \mathbf{x}\mathsf{H}_1(\cdot) + \mathbf{y}\} \in \mathcal{S}$ holds for all $\mathbf{x}, \mathbf{y} \in \beta_{\mathsf{xy}}$ and H_1 as defined in Section 2, where $\mathcal{S} = \{\mathbf{z} \in \mathcal{R}^l : \|\mathbf{z}\|_\infty \leq 2\beta_{\mathsf{c}}\beta_{\mathsf{xy}}\}$.
- *Collision Resistance:* The function family $\{\mathbf{A} : \mathcal{S} \rightarrow \mathcal{R}^n | \mathbf{A} \in \mathcal{H}\}$ is collision resistance where given $\mathbf{A}$, the adversary $\mathcal{A}$ has only a negligible probability to output a collision $(\mathbf{u} \neq \mathbf{u}', \mathbf{A}\mathbf{u} = \mathbf{A}\mathbf{u}')$.
- (ϵ, δ)*-Hiding*: Given $\mathbf{A}, \mathbf{x}, \mathbf{y} \in \beta_{\mathsf{xy}}$ and $h \leftarrow \mathsf{H}_1(m,c)$, let:

$$\Gamma_{\mathbf{A}}(\mathbf{x}, \mathbf{y}, h) = \{\mathbf{x}', \mathbf{y}' \in \beta_{\mathsf{xy}} : \mathbf{A}\mathbf{x} = \mathbf{A}\mathbf{x}' \wedge \mathbf{A}\mathbf{y} = \mathbf{A}\mathbf{y}' \wedge \mathbf{x}h + \mathbf{y} = \mathbf{x}'h + \mathbf{y}'$$

be the set of keys that are compatible with the public key $pk = (\mathbf{t}_c, \mathbf{t}'_c)$ and signature $\mathbf{x}h+\mathbf{y}$ for h as defined above. Then the scheme is (ϵ, δ)-Hiding if $\Pr_{\mathbf{x},\mathbf{y} \in \beta_{\mathsf{xy}}}[\forall h \neq h', |\Gamma_{\mathbf{A}}(\mathbf{x}, \mathbf{y}, h) \cap \Gamma_{\mathbf{A}}(\mathbf{x}, \mathbf{y}, h')| \leq \epsilon |\Gamma_{\mathbf{A}}(\mathbf{x}, \mathbf{y}, h)|] \geq \delta$.
In the following, we use the hiding property where $\epsilon = \frac{1}{2}$ and $\delta = 1$.

Lemma 1. *For he function $\{\mathbf{A}\mathbf{u} : \mathcal{S} \rightarrow \mathcal{R}^n | \mathbf{A} \in \mathcal{R}^{l \times l}\}$ is collision resistance under the average-case Ring-SIS$_{q,n,k,l,\beta_{xy}}$ problem defined in Definition 1.*

Proof. Following the results of [18], if the adversary $\mathcal{A}$ can find two vectors $\mathbf{u}, \mathbf{u}' \in \mathcal{S}$ and $\mathbf{u} \neq \mathbf{u}'$ for a random $\mathbf{A} \in \mathcal{R}^{l \times l}$, such that $\mathbf{A}\mathbf{u} = \mathbf{A}\mathbf{u}'$, then it can find a solution to the average-case Ring-SIS$_{q,n,k,l,\beta_{\mathsf{xy}}}$ problem since $\mathbf{A}(\mathbf{u} - \mathbf{u}') = 0$ and $\|\mathbf{u} - \mathbf{u}'\|_\infty \leq 4\beta_{\mathsf{c}}\beta_{\mathsf{xy}}$. □

Lemma 2. *The closure property holds for the scheme proposed in Algorithm 1.*

Proof.

$$\|\mathbf{x}\mathsf{H}_1(\cdot) + \mathbf{y}\|_\infty \leq \|\mathbf{x}\mathsf{H}_1(\cdot)\|_\infty + \|\mathbf{y}\|_\infty \leq \beta_{\mathsf{c}}\beta_{\mathsf{xy}} + \beta_{\mathsf{c}}\beta_{\mathsf{xy}} = 2\beta_{\mathsf{c}}\beta_{\mathsf{xy}}$$

□

Lemma 3. *([18]) Let n, l, q and β_x be positive integers, $\mathcal{R} = Z[x]/(x^n+1)$ a polynomial ring and suppose that $\beta_x^{ln} > 2^\kappa \beta_0 q^n$. Then for any $\mathbf{A} \in \mathcal{R}^{k \times l}$ and $c \in \beta_c$ we have $Pr_{\mathbf{x},\mathbf{y} \leftarrow \beta_{xy}}[\exists \mathbf{x}', \mathbf{y}' \in \beta_{xy} : \mathbf{A}\mathbf{x} = \mathbf{A}\mathbf{x}' \wedge \mathbf{A}\mathbf{y} = \mathbf{A}\mathbf{y}' \wedge \mathbf{x} \cdot c + \mathbf{y} = \mathbf{x}' \cdot c + \mathbf{y}'] = 1 - 2^{-\kappa}$*

Proof. For any $\mathbf{A} \in \mathcal{H}$, lets consider the function $\{(\mathbf{A}, c) : (\mathbf{x}, \mathbf{y}) \in \mathcal{K} \rightarrow (pk, \mathbf{z})\}$. The domain size of this function is $|\mathcal{K}| = {\beta_{\mathsf{xy}}}^{nl} \cdot {\beta_{\mathsf{xy}}}^{nl}$. By Lemma we have $\mathbf{z} = \mathbf{x}\mathsf{H}_1(\cdot) + \mathbf{y} \leq 2\beta_{\mathsf{c}}\beta_{\mathsf{xy}}$. Therefore, the number of possibilities for $\mathbf{z}$ is (at most) $(4\beta_{\mathsf{c}}\beta_{\mathsf{xy}} + 1)^{nl}$. We also note that there are q^{2n} possibilities for $pk = (t, t')$. For a fixed $\mathbf{A}$, c, and signature, there are at most $((4\beta_{\mathsf{c}}\beta_{\mathsf{xy}} + 1)^{nl})q^n$ possibilities for $(pk, \mathbf{z})$. Therefore, given the results in [18, Lemma 4.1], the probability is $1 - 2^{-\kappa}$ □

Lemma 4. *For $2\beta_{xy} + 1 \geq q^{1/l} \cdot 2^{\kappa/nl}$ and a large q, the scheme satisfies the $(\frac{1}{2})$-Hiding property.*

Proof. By the result of the previous lemma, over the random choice of $\mathbf{x}$ and $\mathbf{y}$, for every message the size of the set $\Gamma(\mathbf{x}, \mathbf{y}, h)$ (for $h \leftarrow \mathsf{H}_1(m,c)$) is at least 2 with probability $1 - 2^{-\kappa}$. We will then show that for all $\mathbf{A}, \mathbf{x}, \mathbf{y}$ and $h \neq h'$, the size of the set $\Gamma(\mathbf{x}, \mathbf{y}, h) \wedge \Gamma(\mathbf{x}, \mathbf{y}, h')$ is at most 1. Next, given the definition of $\Gamma(\cdot)$, we have $\mathbf{x}h + \mathbf{y} = \mathbf{x}'h + \mathbf{y}'$ and $\mathbf{x}h' + \mathbf{y} = \mathbf{x}'h' + \mathbf{y}'$. These relations can be written as $(\mathbf{x} - \mathbf{x}') \cdot (h - h') = 0$. Since $2\beta_{\mathsf{xy}}\beta_{\mathsf{c}}(n\beta_{\mathsf{c}} + 1) < q$ and since $\mathcal{R}$ is an integral domain, we have $\mathbf{x} = \mathbf{x}'$ which implies $\mathbf{y} = \mathbf{y}'$ and this concludes our proof.

□

Therefore, given the results in [18, Theorem 3.2] and the Lemmas above, our scheme proposed in Algorithm 1 is secure in the sense of Definition 6. □

Lemma 5. *The scheme proposed in Algorithm 2 is forward secure.*

Proof. Firstly, to respond to $\texttt{BreakIn}(c)$ queries for the state/period c, $\mathcal{B}$ outputs the full private key of the user $(\mathbf{x}_c, \mathbf{y}_c)$. Note that this is much stronger than returning the private key components or partial signing keys. The scheme proposed in Algorithm 2 utilizes a hash and delete method on the seeds $z_{j,c+1} \leftarrow \mathsf{H}_2(z_{j,c})$ to compute new seeds for each state/period $c+1$ on both the signer and server's side and after the new seed is generated the old will be deleted. Given the properties of the hash function $\mathsf{H}_2(\cdot)$ the provided hash chain ensures that if one (or all) private key components are compromised in time period c, it would be infeasible to compute the private keys prior to state c. Similar methods has also been used in other schemes such as XMSS [33]. □

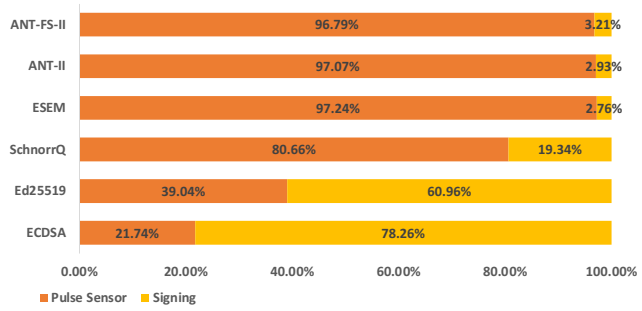

Figure 3: Energy of Signature Generation vs Pulse Sensor

6 PERFORMANCE EVALUATION

In this section we evaluate and compare the performance of ANTand its variants (i.e., ANT-FS and MT-ANT) with some of its closely related counterparts.

As compared to the our base scheme in [18], we significantly reduce the signer memory requirement and communication by not requiring the signer to store and send the public key. For instance, even if we do not consider the hurdle of certificates, the public key size for 138 bits security would be nearly 7.20 KB (for each signature) to be stored at the signer's side and be communicated with each signature. Also note that for a one-time scheme in [18], the signer needs to compute a vector-matrix multiplication to compute the public key for every new signature to be generate. As aforementioned, these performance gains at the signer's side come at no cost for the signer. Instead, given our model, which assumes a computationally capable verifier, the verifier needs to obtain the partial public keys from a set of t distributed servers to form the final public key.

We have fully implemented ANT and its forward secure variant (i.e., ANT-FS) on commodity hardware by utilizing the latest tools that were used in some of the most recent lattice-based candidates in NIST's PQC standardization process. Note that we did not provide the performance analysis of the MT-ANT variant since the signer's computations will be identical to those in ANT. We have also performed a very conservative cost estimation of our schemes on an 8-bit platform.

We compare the performance of our schemes with state-of-the-art digital signature schemes for both of these platforms, in terms of computation, storage and communication.

More specifically, we compare our schemes with both conventional and post-quantum signatures on both commodity hardware and low-end platform (if available). For our lattice-based counterparts, we have selected Dilithium and Bliss.

6.1 Parameters

We set parameters for the R-SIS problem to be hard. Following [5], we set parameters for ANT-II and ANT-III with $\kappa = 103$ and $\kappa = 138$, respectively. Therefore, for both of the variants we set $q = 2^{23} - 2^{13} - 1$, $n = 256$ [5]. Additionally, for ANT-II, we set $\beta_{xy} = 6$, $k = 4$ and $l = 3$ and for For ANT-III, we set $\beta_{xy} = 5$, $k = 5$ and $l = 4$. Given we use the similar tool sets and hard problems as the current lattice-based NIST candidates (e.g., [5]), we believe any future improvements in terms of parameters will contribute to the performance of the new scheme as well.

6.2 Performance on Commodity Hardware

6.2.1 Hardware Configurations. We used a laptop equipped with Intel i7 Kaby Lake Refresh processor @ 1.90 GHz and 16 GB RAM. For our distributed public key servers, we set up Amazon EC2 instances, equipped with an Intel Xeon E5 processor that operates at 2.4 GH. Our servers are located in Oregon.

6.2.2 Libraries. Our implementation is based on PQC NIST candidate [5]. PRF is instantiated with AES in counter mode using Intel intrinsics and used SHAKE256 for hashing. Our implementation is not optimized for any specific platforms and for our counterparts, we used their base reference implementations. Our proof-of-concept implementation is available at https://github.com/Rbehnia/ANT.git.

6.2.3 Experimental Results. Table 2 presents the experimental results of our schemes along with comparison with its post-quantum counterparts.

Signer Computation & Storage: ANT is designed to provide ultra-efficient signing with post-quantum security especially for resource-limited IoT devices (e.g, a microcontroller). Therefore, the signature generation algorithm of ANT, after computing the private key (Steps 2-5 in ANT.Sig(·)), only requires a sparse vector multiplication and one vector addition.

Therefore, as depicted in Table 2, both ANT-II and ANT-III significantly outperform their hash-based counterparts. As compared to one of the most efficient candidates, currently in the third round of NIST PQC standardization [5], our scheme is significantly faster and more memory-efficient on the signer's side. Comparing the signature generation of [5] with ours, firstly, since we do not need to have the rejection sampling step (which could require the repetition of the signing algorithm to 10×) we do not require any repetition of the signing algorithm. Secondly, due to the one-time nature of our scheme, we do not need to compute a commitment in our signing which requires a vector-matrix multiplication. These result in more than 10× faster signature generation in ANT as compared to Dilithium-II and Dilithium-III. In terms of storage and communication, ANT only a 32 byte seed to regenerate the private key components using the Samp functions and for signature size, ANT enjoys from more than 5.5× smaller signature size as compared to Dilithium variants. Moreover, even though explicit side-channel attacks have not been discovered, the concern of such attacks has been studied in [24]. Additionally, while the matrix $\mathbf{A}$ does not need to be stored and can be regenerated via a "*Expand*" function, it still requires memory expansion at the signer side, this memory expansion is up to 14.38 KB for this matrix. Lastly, based on our analytical analysis, the signature size in [24] will be nearly 1.2 KB larger than the one in our scheme for 138 bits security.

Verifier Computation & Storage: Even though ANT and its variants are designed with the focus on providing efficient signature generation, they still enjoy from a rather efficient verification algorithm, as compared with its most efficient counterparts. As alluded to, for each signature verification, ANT requires the verifier to request public key (components) from the t honest-but-curious public key servers, and given our setup, this communication overhead is measured to be on average around 25 *ms* with our EC2 instance in

Table 2: Experimental performance comparison of ANT and its post-quantum counterparts on a commodity hardware

Scheme	Signing Cycles	Private Key† (KB)	Signature (KB)	Verification Cycles	Public Key (KB)	Rejection/Gaussian (Sampling)
SPHINCS+ [14]	222, 503, 189	0.06	17, 088	14, 681, 407	0.03	×
XMSS-MT [15]	8, 075, 528	5.86	4.85	2, 490, 844	0.06	×
Dilithium-II [5]	865, 921	2.288	2.42	216, 654	1.312	✓
Dilithium-III [5]	1, 387, 776	3.184	3.293	348, 640	1.952	✓
ANT-II	81, 333	0.03	0.432	293, 184	5.89	×
ANT-III	84, 805	0.03	0.56	399, 214	7.36	×
ANT-FS-II	82, 829	0.09	0.432	293, 184	5.89	×
ANT-FS-III	86, 914	0.09	0.56	399, 214	7.36	×

† For SPHINCS+ we use the parameters $n = 16$, $h = 66$, $d = 22$, $b = 6$, $k = 33$, $w = 16$ for 128-bits security. For XMSS-MT, we benchmarked the *XMSSMT-SHA2_20/2_256* variant. Dilithium-II and Dilithium-III provide similar security to our ANT-II and ANT-III variants. We consider $t = 3$ (i.e., three servers setting) which affects the private key size in our forward secure variant. Note that for ANT the verification cycles does not include the network delay to receive partial public keys which is around 25 ms per server in our setting. Also, the public size in ANT is per-signature.

Oregon. We also note that as opposed to Dilithium, the public key storage overhead in our scheme is per signature. We stress that given our system model and performance gain on the signer we believe this is a worthy trade-off.

6.3 Performance on microcontrollers

6.3.1 Hardware Configurations. IoT systems employ a plethora of embedded and resource constraint processors mainly to minimize the size, power consumption and cost. In one line, ARM Cortex processors gain popularity. On the another line, truly embedded devices with popular microcontroller are often 32-,16- or even 8-bit have been considered in various IoT applications [35]. To ensure that our proposed system meets the most stringent requirement of these systems, we selected AVR ATmega 2560 microcontroller as our embedded device mostly since it is adopted in practice, especially for medical devices [6]. However, we expect its performance benefits to apply to other class of IoT devices such as ARM Cortex M4 and others such 32-bit and 16-bit microcontrollers. AVR ATmega 2560 is an 8- bit microcontroller with 8 KB SRAM, 256 KB flash memory, 4 KB EEPROM and maximum clock speed is 16 MHz. Our AVR ATmega 2560 operates at $5V$ voltage level and takes 20 mA of current. It is powered by a 2200 mAh power-pack. We measured the energy with th formula $E = V \times I \times t$, where I is the amperage/current and t is the computation time. With the aim of measuring the energy overhead of our proposed systems on low-end IoT devices, we measured the energy consumption of a pulse sensor [36] compatible with the AVR ATmega 2560.

6.3.2 Libraries. We recommend ANT-II for deployment on 8-bit microcontroller due to its smaller parameter sizes. We have selected our counterparts with a comparable security level. We used CHACHA20 stream cipher [37] for random generation functions due to its high efficiency. We simulated our hash functions with BLAKE2s [38] since it is specially optimized for low-end devices.

6.3.3 Experimental Results. Table 1 and Figure 3 presents the experimental results and energy consumption of our schemes along with comparison with its post-quantum counterparts.

To our knowledge, there is no 8-bit implementation of Dilithium available. Therefore, we have selected BLISS [21] as our post-quantum counterpart. BLISS and Dilithium are both based on the "Fiat-Shamir with Aborts" framework [20]. However, BLISS is based on NTRU (as opposed to LWE and SIS) and is known to have enjoyed from a faster signature generation than Dilithium. However, note that BLISS, while being faster than Dilithium, is not considered in NIST PQC standardization process. One of the reasons could be the lack of worst case reduction for the NTRU problem as opposed to other lattice-based problems like the SIS. Another potential candidate in the current NIST standardization process is Falcon [12]. Unlike BLISS and Dilithium, Falcon is instantiated over NTRU lattices and is based on the hash-and-sign paradigm with a trapdoor sampler called "fast Fourier sampling". Falcon is known to be more RAM efficient, however, in terms of signing speed, since Falcon relies on 64-bit floating point arithmetic, which is not natively available for low-end devices (e.g., 8-bit processors), and has to be emulated, it would become much slower than its FSA-based counterparts [39]. To provide a reference, the performance results reported in [40] on an ARM Cortex-M at 168 MHz, the signing on Dilithium-III takes 8,348,349 cycles, while Falcon-I takes 80,503,242, that is nearly 10× slower for a lower security level . For a compete signing speed analysis between NIST candidates please see [40]. We also compare ANT with a widely adopted classical scheme.

Singer Computation: Among our classical counterparts, ESEM [10] performance is nearly comparable to our schemes. However, we have included ESEM for the sake of completeness and note the method that ESEM utilizes (BPV method [41]) to achieve this high efficiency is based on the hidden subset sum problem which was recently shown to be susceptible to a polynomial time attack [**?**]. Therefore, to account for the security loss resulted from the attack, ESEM parameters should be updated, which could result in a less efficient scheme. However, the numbers represented here are based on the original parameters. After ESEM, ANT-II and ANT-FS-II are 8× and 7× faster than our closet classical counterpart (i.e., SchnorrQ [42]). As depicted in 1, ANT-II and ANT-FS-II signature generation are 16× and 13× faster than that in BLISS [11]. Similar to our classical counterparts, ANT-II has an optimal private key size of 32 bytes only. However, as stated in Section 4.2, the private key size in ANT-FS is linear to t - the number of public key generation servers. The signature size in ANT is 30% smaller than that in BLISS. However, note that since the signer is not involved in computing, storing or communicating the public keys, the larger public key size in ANT(and its variants) does not affect the performance of the signer in any way.

Energy Consumption of the Signer: As for the energy consumption of ANT (and its variants), as presented in Figure 3, with a 2200 mAh battery ANT-II and ANT-FS-II can generate approximately 745,000 and 675,000 signatures. With its current (insecure) parameters, this number is slightly higher at 798, 000 for ESEM. We estimated this number to be at least one order of magnitude smaller (compared to ANT) for BLISS [11].

We compared the energy overhead of our schemes, and their classical counterparts, on a pulse sensor (with potential medical applications). We considered the sensor access time interval (read) to be every 10 seconds. Given its data sheet, the sensor draws $4.5mA$ of current at $3\ V$. Our 8-bit microcontroller (i.e., ATmega2560) takes 1 ms to read from the sensor. While running, it takes $5V \times 20mA \times 1ms$, and μA in standby (power-saving) mode. As depicted in 3, after ESEM, which is not secure with its current parameters, ANT-II and ANT-FS-II minimize the energy over head and only consume 2.93% and 3.21% of energy (as compared to the pulse sensor), respectively. This is significantly lower than its closet counterpart SchnorrQ [42], which consumes around 19% of the energy. Given the microcontroller type (operating at 32 MHz) and the signing cycles, we anticipate the BLISS would also have a much higher energy consumption overhead as compared to our schemes. Therefore, our experiments confirm that ANT (and its variants) significantly reduce the energy consumption overhead on low-end IoT devices.

Side-Channel Resiliency: To our knowledge, the signing algorithm of ANT and its variants is not susceptible to side-channel attacks that are specific to some lattice-based constructions. For instance, since the singing algorithm of ANT and its variants does not require any Gaussian sampling or rejection sampling it is not susceptible to attacks targeting [22–24] these operations. We note that these attack can be addressed with the cost of additional computation and/or communication overhead.

Acknowledgment

The work of Attila A. Yavuz is supported by the NSF CAREER Award CNS-1917627 and an unrestricted gift via Cisco Research Award.

REFERENCES

[1] C. Camara, P. Peris-Lopez, and J. E. Tapiador, "Security and privacy issues in implantable medical devices: A comprehensive survey," *Journal of Biomedical Informatics*, vol. 55, pp. 272 – 289, 2015.

[2] Y. Chen, W. Xu, L. Peng, and H. Zhang, "Light-weight and privacy-preserving authentication protocol for mobile payments in the context of iot," *IEEE Access*, vol. 7, pp. 15 210–15 221, 2019.

[3] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM Review*, vol. 41, no. 2, pp. 303–332, 1999.

[4] *ANSI X9.62-1998: Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)*, American Bankers Association, 1999.

[5] L. Ducas, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehle, "Crystals – dilithium: Digital signatures from module lattices," Cryptology ePrint Archive, Report 2017/633, 2017, http://eprint.iacr.org/2017/633.

[6] M. Rushanan, A. D. Rubin, D. F. Kune, and C. M. Swanson, "Sok: Security and privacy in implantable medical devices and body area networks," in *Proceedings of the 2014 IEEE Symposium on Security and Privacy*, ser. SP '14. IEEE Computer Society, 2014, pp. 524–539.

[7] K. MacKay, "micro-ecc: Ecdh and ecdsa for 8-bit, 32-bit, and 64-bit processors," Github Repository, 2013. [Online]. Available: https://github.com/kmackay/micro-ecc

[8] M. Hutter and P. Schwabe, "Nacl on 8-bit avr microcontrollers," in *Progress in Cryptology – AFRICACRYPT 2013*, A. Youssef, A. Nitaj, and A. E. Hassanien, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 156–172.

[9] Z. Liu, P. Longa, G. C. C. F. Pereira, O. Reparaz, and H. Seo, "Four$\mathbb{Q}$ on embedded devices with strong countermeasures against side-channel attacks," in *Cryptographic Hardware and Embedded Systems – CHES 2017*, W. Fischer and N. Homma, Eds. Cham: Springer International Publishing, 2017, pp. 665–686.

[10] M. O. Ozmen, A. A. Yavuz, and R. Behnia, "Energy-aware digital signatures for embedded medical devices," in *2019 IEEE Conference on Communications and Network Security (CNS)*, 2019, pp. 55–63.

[11] T. Güneysu, V. Lyubashevsky, and T. Pöppelmann, "Practical lattice-based cryptography: A signature scheme for embedded systems," in *Cryptographic Hardware and Embedded Systems – CHES 2012*, E. Prouff and P. Schaumont, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 530–547.

[12] P.-A. Fouque, J. Hoffstein, P. Kirchner, V. Lyubashevsky, T. Pornin, T. Prest, T. Ricosset, G. Seiler, W. Whyte, and Z. Zhang, "Falcon: Fast-fourier lattice-based compact signatures over ntru," *Submission to the NIST?s post-quantum cryptography standardization process*, 2018.

[13] J. Ding and D. Schmidt, "Rainbow, a new multivariable polynomial signature scheme," in *International Conference on Applied Cryptography and Network Security*. Springer, 2005, pp. 164–175.

[14] D. J. Bernstein, A. Hülsing, S. Kölbl, R. Niederhagen, J. Rijneveld, and P. Schwabe, "The sphincs⁺ signature framework," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 2129?2146.

[15] A. Hülsing, L. Rausch, and J. Buchmann, "Optimal parameters for xmssmt," in *Security Engineering and Intelligence Informatics*, A. Cuzzocrea, C. Kittl, D. E. Simos, E. Weippl, and L. Xu, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 194–208.

[16] A. Hülsing, J. Rijneveld, and P. Schwabe, "Armed sphincs," in *Public-Key Cryptography–PKC 2016*. Springer, 2016, pp. 446–470.

[17] J. W. Bos, A. Hülsing, J. Renes, and C. van Vredendaal, "Rapidly verifiable xmss signatures," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 137–168, 2021.

[18] V. Lyubashevsky and D. Micciancio, "Asymptotically efficient lattice-based digital signatures," *J. Cryptology*, vol. 31, no. 3, pp. 774–797, 2018.

[19] A. A. Yavuz, "Eta: efficient and tiny and authentication for heterogeneous wireless systems," in *Proceedings of the sixth ACM conference on Security and privacy in wireless and mobile networks*, ser. WiSec '13. New York, NY, USA: ACM, 2013, pp. 67–72.

[20] V. Lyubashevsky, "Fiat-shamir with aborts: Applications to lattice and factoring-based signatures," in *Advances in Cryptology – ASIACRYPT 2009: 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings*, M. Matsui, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 598–616.

[21] L. Ducas, A. Durmus, T. Lepoint, and V. Lyubashevsky, "Lattice signatures and bimodal gaussians," in *Advances in Cryptology – CRYPTO 2013: 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, R. Canetti and J. A. Garay, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 40–56.

[22] L. Groot Bruinderink, A. Hülsing, T. Lange, and Y. Yarom, "Flush, gauss, and reload – a cache attack on the bliss lattice-based signature scheme," in *Cryptographic Hardware and Embedded Systems – CHES 2016: 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings*, B. Gierlichs and A. Y. Poschmann, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 323–345.

[23] T. Espitau, P. Fouque, B. Gérard, and M. Tibouchi, "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, CCS 2017*, 2017, pp. 1857–1874.

[24] V. Migliore, B. Gérard, M. Tibouchi, and P. Fouque, "Masking dilithium - efficient implementation and side-channel evaluation," in *Applied Cryptography and Network Security - 17th International Conference, ACNS 2019, Bogota, Colombia, June 5-7, 2019, Proceedings*, 2019, pp. 344–362.

[25] C. Pereida García, B. B. Brumley, and Y. Yarom, ""make sure dsa signing exponentiations really are constant-time"," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. New York, NY, USA: ACM, 2016, pp. 1639–1650.

[26] H. N. Noura, O. Salman, A. Chehab, and R. Couturier, "Distlog: A distributed logging scheme for iot forensics," *Ad Hoc Networks*, vol. 98, p. 102061, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1570870519306997

[27] E. Kiltz, V. Lyubashevsky, and C. Schaffner, "A concrete treatment of fiat-shamir signatures in the quantum random-oracle model," in *Advances in Cryptology – EUROCRYPT 2018*, J. B. Nielsen and V. Rijmen, Eds. Cham: Springer International Publishing, 2018, pp. 552–586.

[28] Y. Ishai and E. Kushilevitz, "Improved upper bounds on information-theoretic private information retrieval (extended abstract)," in *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing*, ser. STOC '99. New

York, NY, USA: Association for Computing Machinery, 1999, p. 79?88. [Online]. Available: https://doi.org/10.1145/301250.301275

[29] M. Bellare and P. Rogaway, "The security of triple encryption and a framework for code-based game-playing proofs," in *Advances in Cryptology - EUROCRYPT 2006*, S. Vaudenay, Ed. Springer Berlin Heidelberg, 2006, pp. 409–426.

[30] M. Abdalla and L. Reyzin, "A new forward-secure digital signature scheme," in *Advances in Cryptology — ASIACRYPT 2000*, T. Okamoto, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 116–129.

[31] L. Reyzin and N. Reyzin, "Better than BiBa: Short one-time signatures with fast signing and verifying," in *Proceedings of the 7th Australian Conference on Information Security and Privacy (ACIPS '02)*. Springer-Verlag, 2002, pp. 144–153.

[32] D. J. Bernstein, D. Hopwood, A. Hülsing, T. Lange, R. Niederhagen, L. Papachristodoulou, M. Schneider, P. Schwabe, and Z. Wilcox-O'Hearn, "SPHINCS: Practical stateless hash-based signatures," in *Advances in Cryptology – EUROCRYPT 2015: 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer Berlin Heidelberg, April 2015, pp. 368–397.

[33] J. Buchmann, E. Dahmen, and A. Hülsing, "Xmss - a practical forward secure signature scheme based on minimal security assumptions," in *Proceedings of the 4th International Conference on Post-Quantum Cryptography*, ser. PQCrypto'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 117–129.

[34] R. C. Merkle, "A certified digital signature," in *Proceedings on Advances in cryptology*, ser. CRYPTO '89. New York, NY, USA: Springer-Verlag, 1989, pp. 218–238.

[35] D. Atkins, "Requirements for post-quantum cryptography on embedded devices in the iot."

[36] "Pulse sensor by world famous electronics llc." https://pulsesensor.com.

[37] D. J. Bernstein, "New stream cipher designs," M. Robshaw and O. Billet, Eds. Berlin, Heidelberg: Springer-Verlag, 2008, ch. The Salsa20 Family of Stream Ciphers, pp. 84–97.

[38] J.-P. Aumasson, L. Henzen, W. Meier, and R. C.-W. Phan, "Sha-3 proposal blake," Submission to NIST (Round 3), 2010. [Online]. Available: http://131002.net/blake/blake.pdf

[39] M. J. Kannwischer, J. Rijneveld, P. Schwabe, and K. Stoffelen, "pqm4: Testing and benchmarking nist pqc on arm cortex-m4," 2019.

[40] A. Khalid, S. McCarthy, M. O?Neill, and W. Liu, "Lattice-based cryptography for iot in a quantum world: Are we ready?" in *2019 IEEE 8th International Workshop on Advances in Sensors and Interfaces (IWASI)*, 2019, pp. 194–199.

[41] V. Boyko, M. Peinado, and R. Venkatesan, "Speeding up discrete log and factoring based schemes via precomputations," in *Advances in Cryptology — EUROCRYPT'98: International Conference on the Theory and Application of Cryptographic Techniques Espoo, Finland, May 31 – June 4, 1998 Proceedings*. Springer Berlin Heidelberg, 1998, pp. 221–235.

[42] C. Costello and P. Longa, "Schnorrq: Schnorr signatures on fourq," MSR Tech Report, 2016. Available at: https://www. microsoft. com/en-us/research/wp-content/uploads/2016/07/SchnorrQ. pdf, Tech. Rep., 2016.