Lightweight and Breach-Resilient Authenticated Encryption Framework for Internet of Things

Saif Eddine Nouma, Attila A. Yavuz

University of South Florida, Tampa, FL, USA
Email: {saifeddinenouma, attilaayavuz}@usf.edu

Abstract—The Internet of Things (IoT) relies heavily on resource-limited devices to communicate critical (e.g., military data) information under low-energy adversarial environments and low-latency wireless channels. Authenticated Encryption (AE) guarantees confidentiality, authenticity, and integrity, making it a vital security service for IoT. However, current deployed (lightweight) AE standards lack essential features like key compromise resiliency and compact authentication tags, as well as performance enhancements such as offline-online cryptography. To address these gaps, we propose `Graphene`, the first (to our knowledge) symmetric Forward-secure and Aggregate Authenticated Encryption (FAAE) framework designed for the performance and security demands of low-end IoT infrastructures. `Graphene` innovates by synergizing key evolution strategies and offline-online cryptographic processing with Universal Message Authentication Codes (UMACs) to guarantee breach-resiliency, near-optimal online latency, and compactness. We demonstrate `Graphene`'s efficiency through two distinct instantiations, each balancing unique performance trade-offs with extensibility for diverse MACs. Our experimental evaluation on commodity hardware and 32-bit ARM Cortex-M4 microcontroller shows `Graphene`'s significant performance gains over existing alternatives. `Graphene` is also backward compatible with standard-compliant cryptographic implementations. We release our implementation as open source for public testing and adaptation.

Index Terms—Lightweight Cryptography, AE, Wireless IoT

I. INTRODUCTION

The rapid expansion of wireless IoT systems across critical domains like smart healthcare, battlefield surveillance, and industrial automation has made secure communication a critical necessity [1], [2], [3], [4]. Resource-constrained IoT devices, such as medical pacemakers, aerial drones, and surveillance cameras, continuously generate sensitive telemetry and offload it to an edge or cloud server for remote diagnosis and real-time alerts [5]. However, ensuring secure and efficient wireless communication remains challenging due to the inherent resource constraints of embedded devices, including limited computational power, bandwidth, and battery life.

Authenticated Encryption (AE) [6] combines symmetric encryption with Message Authentication Codes (MACs) to provide confidentiality, integrity, and authenticity, making it a compelling solution for wireless IoT infrastructures. While existing AE standards (e.g., AES-GCM [7]) achieve high efficiency through hardware acceleration across various platforms [8], they fall short of fully utilizing advanced cryptographic techniques to meet these rigorous security and performance

demands of constrained IoT networks: *(1) Offline-Online (OO) cryptography* to enable low-latency and energy-efficient processing [1], [3]. *(2) Integration of Universal MACs within AEs*, leveraging partial homomorphism [9], [10] for efficient batch verification of authentication tags. *(3) Forward security* to mitigate the impact of key compromise attacks [2], [11]. *(4) Tag aggregation* to reduce transmission and storage overhead, enhancing overall system efficiency [12], [13]. In the following sections, we review related work on AE and MAC schemes, with a particular emphasis on these essential properties.

A. Related Work and Limitations

ASCON [14], the NIST lightweight AE standardization winner, lacks properties (1)-(4) due to its integrated AE structure. Specifically, its encryption precludes ciphertext-independent preprocessing while tag verification requires full payload decryption, making it incompatible with desirable efficient batch verification. It is also vulnerable to preimage attacks [14], which significantly constrain its use in critical domains.

AES-GCM [3] is a widely used AE standard in protocols like DTLS 1.3 and IEEE 802.15.4 for constrained sensor and low-rate wireless personal area networks. ChaCha20-Poly1305 [6] is also used in TLS 1.3 for IoT chips (e.g., ARM Cortex-M) when hardware-accelerated AES is not available. Both share the same design principle: a counter mode of operation (CTR) for encryption, while authentication is based on a Wegman-Carter MAC construction [6] from polynomial hash functions.

Few encryption works explore precomputation (1) of the counter mode, leveraging idle periods of low-end devices. For instance, [3] analyzes latency in industrial applications and proposes precomputed block cipher templates.

Wegman-Carter MAC constructions [6] are more efficient than traditional Hash-based MACs (HMACs) [2]. They offer precomputation (1) due to nonce-based structure. To our knowledge, none have explored AE integration with desirable properties (1-4). GHASH and Poly1305, as authenticators of AES-GCM and ChaCha20-Poly1305 provide (1) and amenable to key update (3) and tag aggregation (4). [6] proposes variants of Poly1305 with security and performance trade-offs. [1] supports (1) but is efficient only for small inputs ($\approx$ 1-4 bytes), while including forward security is costly due to large pre-stored key tables. Some MAC variants based on universal hash functions [9], [10] provide efficient batch verification (2).

Forward-secure and aggregate MACs [11], [2] achieve compact aggregate tags and key compromise resiliency but omits precomputation (1) along with an integrated precomputable encryption. Progressive MACs [15] offers short authentication tags and provides resiliency against synchronization attacks, but with tricky internal state management and lacks (2-4).

Overall, there is a gap in the state of the art in achieving lightweight, breach-resilient, precomputable, and compact authenticated encryption for resource-constrained IoT systems.

B. Our Contributions

We propose `Graphene`, to our knowledge, the first comprehensive breach-resilient and compact AE framework to enhance resiliency and online performance via forward-security, pre-computation, partial homomorphism with batch processing, and various aggregation modes. We outline the desirable properties of `Graphene` as follows:

● *A New Forward-secure and Aggregate AE Framework (FAAE)*: `Graphene` harnesses key evolution with sequential tag aggregation considering special MACs and combination modes. `Graphene` uncovers synergies among forward-secure encryption and UMACs overlooked by previous approaches while being backward compatible with current standard implementations. Therefore, it achieves high efficiency and security while opening a path for alternative constructions through generalizable integrations and ease of implementation.

● *High Computation Efficiency via OO Cryptography*: OO cryptography shifts input-independent cryptographic computations to the pre-processing stage during idle periods, reducing online latency. Although studied for digital signatures, OO methods have not been explored for FAAE. `Graphene` explores the synergies of various UMACs and AE modes in OO settings and achieves a significant online performance gain with only a modest increase in memory consumption. For example, our `Graphene-Poly` instantiation with OO is $28\times$ *faster than* FAAE in a batch of 1024 16-byte telemetry on a 32-bit Cortex-M4 embedded processor, with only 32KB extra storage (12% of available SRAM). In wearable medical settings, this results in significant savings in battery life during online operations, as the next batch of OO keys can be supplied when the device is recharged while telemetry is uploaded. In aerial drone applications, latency reduction can improve flight safety and real-time telemetry transmission (where key storage is a lesser concern). Finally, beyond benefiting resource-limited devices, fast online operations are advantageous for receivers (e.g., edge clouds) that may need to verify and decrypt millions of telemetry packets simultaneously from various senders.

● *Adaptable Performance Choices and Tunable Security with* `Graphene` *Instantiations*: `Graphene` instantiations integrate well-established standards and MACs, offering enhanced performance and security features: *(i)* `Graphene-AE` provides a standard security level (128-bit) and compliance, operating $3\times$ faster than basic FAAE with an additional 16KB storage on 16-byte inputs with a single-key initialization. *(ii)* `Graphene-Poly` delivers high efficiency with medium 103-bit security, running $8.1\times$ faster than `Graphene-AE` with the same 16KB storage requirement. `Graphene` supports aggregation in various modes, including cryptographic hash functions and XORing, each providing distinct properties.

● *Full-Fledged Implementation and Comprehensive Benchmarks*: We implemented `Graphene` on commodity hardware (x86/64) and a low-end microcontroller (32-bit ARM Cortex-M4). Our performance results highlight the efficiency of each instantiation, guiding its application context accordingly. To encourage reproducibility, we release our implementation at:

https://github.com/saifnouma/Graphene

II. PRELIMINARIES AND MODELS

Notation. $||$ and $|x|$ denote concatenation and bit length of variable x, respectively. $x \xleftarrow{\$} \mathcal{S}$ denotes x is randomly selected from the finite set $\mathcal{S}$ using a uniform distribution. $\{0,1\}^*$ denotes a set of binary strings of any finite length. $\mathbb{Z}_q$ denotes the finite field of prime order q. $\boldsymbol{x}$ denotes a set of items $\{x_1, \ldots, x_n\}$, where $n = |\boldsymbol{x}|$. $\oplus$ denotes XOR operation. Add_q denotes modular addition over modulus q. PRF is a keyed pseudo-random function. It is secure if its output ($\text{PRF}_k(m)$) is indistinguishable from a random string. H is a collision-resistant one-way cryptographic hash function.

Universal MACs. [9] consist of a Carter-Wegman construction with a universal hash family for efficient data integrity. Formally, $\text{MAC}_k(m, n) = F_r(m) + \text{PRF}_s(n)$ where $k \leftarrow (r, s)$ is the private key, F is a universal hash family and $\text{PRF}_s(n)$ is precomputable. m and n are the message and a counter, respectively. The security of universal MACs depends on the security of PRF and F. F is secure if it is ϵ-almost universal hash function, i.e., $Pr[F_r(m) - F_r(m') = \delta] = \epsilon$ where (m, m') are distinct messages and δ in the range of F.

System Model. It consists of two main entities, as depicted in Fig. 1: *(1) Resource-constrained IoT devices (e.g., medical implants)*: operate in adversarial environments under constrained resources of storage, memory, and computation. They continuously generate sensitive data and then periodically upload the telemetry to a nearby edge server. This storage-and-forward setting has vast applications in IoT, such as digital twins and wearable medical devices [5], [16]. *(2) Resourceful verifier*: is a storage and computation resourceful entity, being a patient's phone or a clinical gateway in IoMT, or a nearby access point in IoBT. It receives encrypted and authenticated data from low-end IoT devices, verifies and decrypts them, and optionally offloads to a cloud server for diagnosis and actuation. The two main entities pre-share a private key.

Threat Model. We assume a resourceful probabilistic polynomial-time (PPT) adversary $\mathcal{A}$ capable of launching: *(1) Passive attacks* monitor and analyze encrypted and authenticated traffic. *(2) Active attacks* manipulate encrypted packets during transmission. *(3) Key compromise attacks* extract the long-term symmetric private keys during a system breach (e.g. malware) or a side-channel attack. *(4) Mix-and-match attacks* target systems that mix multiple aggregate tags across multiple
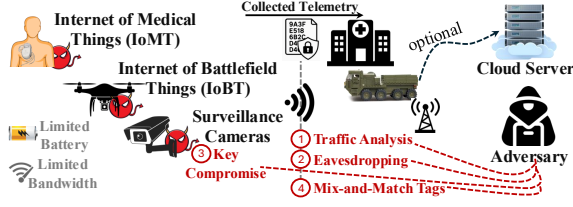
Fig. 1: Our system and threat models

users to forge an aggregate MAC scheme [13]. The adversary aims to decrypt previously intercepted traffic using compromised private keys and produce a forgery on the aggregate and forward-secure authentication tag with the stored telemetry.

Security Model. Our security model is based on the security of its underlying (authenticated) encryption and MAC components. We consider forward security, which implies periodic key updates, an essential feature in adversarial IoT environments against key compromise attacks. For confidentiality, `Graphene` adopts forward-secure indistinguishability under chosen plaintext attacks (F-IND-CPA) [7]. For MACs, we follow the security of previous FAAEs [2], [11], with forward-secure existential unforgeability against chosen message attacks (F-EU-CMA) [17]. F-EU-CMA extends to FA-EU-CMA to cover security against mix-and-match attacks against aggregate MACs [13]. These notions collectively ensure that even an adaptive PPT adversary $\mathcal{A}$ can neither decrypt previously and currently encrypted telemetry traffic nor forge valid aggregate and forward-secure tags without detection. $\mathcal{A}$ (t, ϵ, q_s)-break denotes $\mathcal{A}$ can break a cryptographic scheme in time t with probability ϵ and at most q_s queries.

III. PROPOSED SCHEMES

A standard FAAE scheme typically applies conventional encryption with an HMAC. Its aggregation and forward security mechanisms are built on only a standard hash function. However, this construction falls short of achieving the high efficiency and desirable properties outlined in Section I, such as precomputation OO capabilities for minimal online latency, partial homomorphic features, and flexible aggregation modes, all essential for our system model. While prior MACs have explored precomputation, these efforts were isolated, neglecting key evolution and encryption. In this work, we introduce `Graphene`, the first FAAE framework designed to deliver near-optimal online computational efficiency and compactness, tailored for resource-constrained IoT environments.

A. Generic `Graphene` Framework

Fig. 2 presents the algorithmic description of `Graphene`.

The key generation (`Graphene.Kg`) algorithm takes batch size n as the number of messages to be processed and four flags, first two indicate `ENC` and `MAC` support for `OO`. For example, AES-CTR and Poly1305 [6] support precomputation with `ENC-OO` and `MAC-OO` properties. `Graphene.Kg` outputs initial private key(s) sk_1 based on the selected primitives.
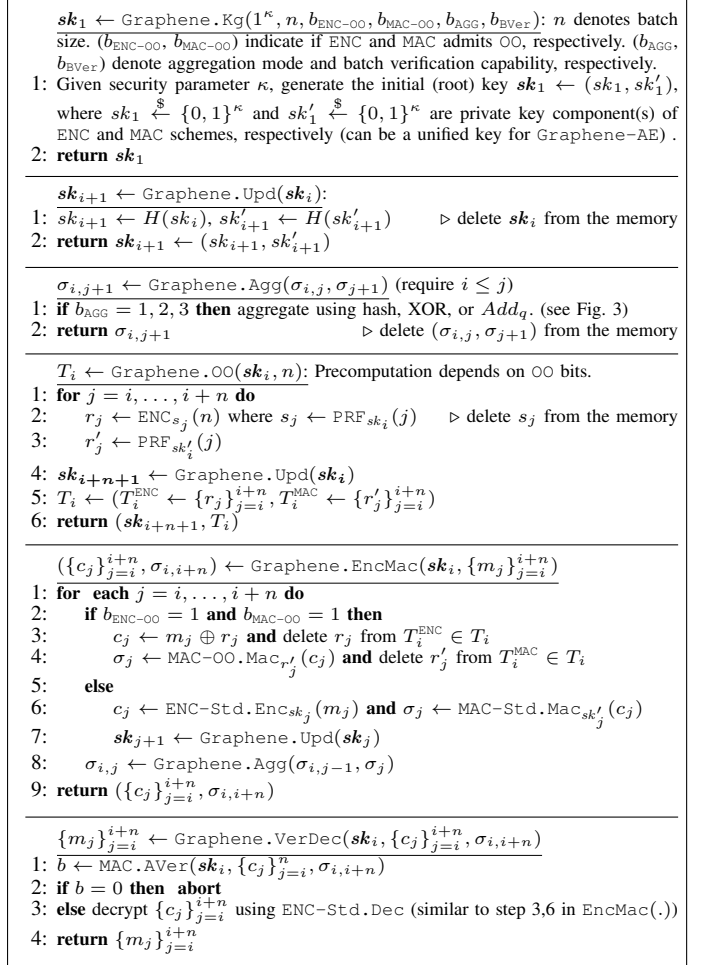
$\underline{sk_1 \leftarrow \text{Graphene.Kg}(1^\kappa, n, b_{\text{ENC-OO}}, b_{\text{MAC-OO}}, b_{\text{AGG}}, b_{\text{BVer}})}$: n denotes batch size. $(b_{\text{ENC-OO}}, b_{\text{MAC-OO}})$ indicate if `ENC` and `MAC` admits `OO`, respectively. $(b_{\text{AGG}}, b_{\text{BVer}})$ denote aggregation mode and batch verification capability, respectively.
1: Given security parameter κ, generate the initial (root) key $sk_1 \leftarrow (sk_1, sk_1')$, where $sk_1 \xleftarrow{\$} \{0,1\}^\kappa$ and $sk_1' \xleftarrow{\$} \{0,1\}^\kappa$ are private key component(s) of `ENC` and `MAC` schemes, respectively (can be a unified key for `Graphene-AE`).
2: **return** sk_1

$\underline{sk_{i+1} \leftarrow \text{Graphene.Upd}(sk_i)}$:
1: $sk_{i+1} \leftarrow H(sk_i), sk_{i+1}' \leftarrow H(sk_{i+1}')$ ▷ delete sk_i from the memory
2: **return** $sk_{i+1} \leftarrow (sk_{i+1}, sk_{i+1}')$

$\underline{\sigma_{i,j+1} \leftarrow \text{Graphene.Agg}(\sigma_{i,j}, \sigma_{j+1}) \text{ (require } i \leq j)}$
1: **if** $b_{\text{AGG}} = 1, 2, 3$ **then** aggregate using hash, XOR, or Add_q. (see Fig. 3)
2: **return** $\sigma_{i,j+1}$ ▷ delete $(\sigma_{i,j}, \sigma_{j+1})$ from the memory

$\underline{T_i \leftarrow \text{Graphene.OO}(sk_i, n)}$: Precomputation depends on `OO` bits.
1: **for** $j = i, \dots, i + n$ **do**
2: $\quad r_j \leftarrow \text{ENC}_{s_j}(n)$ where $s_j \leftarrow \text{PRF}_{sk_i}(j)$ ▷ delete s_j from the memory
3: $\quad r_j' \leftarrow \text{PRF}_{sk_i'}(j)$
4: $sk_{i+n+1} \leftarrow \text{Graphene.Upd}(sk_i)$
5: $T_i \leftarrow (T_i^{\text{ENC}} \leftarrow \{r_j\}_{j=i}^{i+n}, T_i^{\text{MAC}} \leftarrow \{r_j'\}_{j=i}^{i+n})$
6: **return** (sk_{i+n+1}, T_i)

$\underline{(\{c_j\}_{j=i}^{i+n}, \sigma_{i,i+n}) \leftarrow \text{Graphene.EncMac}(sk_i, \{m_j\}_{j=i}^{i+n})}$
1: **for each** $j = i, \dots, i + n$ **do**
2: $\quad$ **if** $b_{\text{ENC-OO}} = 1$ **and** $b_{\text{MAC-OO}} = 1$ **then**
3: $\quad\quad c_j \leftarrow m_j \oplus r_j$ and delete r_j from $T_i^{\text{ENC}} \in T_i$
4: $\quad\quad \sigma_j \leftarrow \text{MAC-OO.Mac}_{r_j'}(c_j)$ **and** delete r_j' from $T_i^{\text{MAC}} \in T_i$
5: $\quad$ **else**
6: $\quad\quad c_j \leftarrow \text{ENC-Std.Enc}_{sk_j}(m_j)$ **and** $\sigma_j \leftarrow \text{MAC-Std.Mac}_{sk_j'}(c_j)$
7: $\quad\quad sk_{j+1} \leftarrow \text{Graphene.Upd}(sk_j)$
8: $\quad \sigma_{i,j} \leftarrow \text{Graphene.Agg}(\sigma_{i,j-1}, \sigma_j)$
9: **return** $(\{c_j\}_{j=i}^{i+n}, \sigma_{i,i+n})$

$\underline{\{m_j\}_{j=i}^{i+n} \leftarrow \text{Graphene.VerDec}(sk_i, \{c_j\}_{j=i}^{i+n}, \sigma_{i,i+n})}$
1: $b \leftarrow \text{MAC.AVer}(sk_i, \{c_j\}_{j=i}^{n}, \sigma_{i,i+n})$
2: **if** $b = 0$ **then abort**
3: **else** decrypt $\{c_j\}_{j=i}^{i+n}$ using `ENC-Std.Dec` (similar to step 3,6 in `EncMac(.)`)
4: **return** $\{m_j\}_{j=i}^{i+n}$

Fig. 2: Generic `Graphene` Framework

We investigate different aggregation algorithms for `Graphene.Agg`: *(1) Hash-based Accumulator*: computes a digest using a cryptographic hash function. It is an immutable aggregation (i.e., $\sigma \leftarrow H(\sigma_1 || \sigma_2)$). *(2) Bitwise XOR*: is an efficient mutable aggregation using XOR operations proportional to the tag size ($\sigma \leftarrow \sigma_1 \oplus \sigma_2$).

The authenticated encryption (`Graphene.EncMac`) algorithm executes `Graphene.OO` if `OO` flags are enabled before receiving telemetry. The step 2 in `OO` precomputes a cipher stream of the same size as the input. `OO` can be made input-independent by storing n with the maximum possible input size. During the online phase, `EncMac` either retrieves the precomputed encryption stream r_j and performs an XOR per input size (step 3) or uses `ENC-Std` (step 6). Similarly, the authentication tag σ_j is generated via `MAC-OO` (step 4) according to `MAC` flag ($b_{\text{MAC-OO}}$) or `MAC-Std` (step 6). After each tag generation, `Graphene.EncMac` deletes the pre-ciphertexts and one-time keys (step 3-4), and finally outputs the ciphertexts $\{c_j\}_{j=i}^{i+n}$ and aggregate tag $\sigma_{i,i+n}$ per batch.

The decryption algorithm (`Graphene.VerDec`) consists of verifying $\sigma_{i,i+n}$ using aggregate verification (`MAC.AVer`) where batch verification is used if b_{BVer} is enabled. Certain
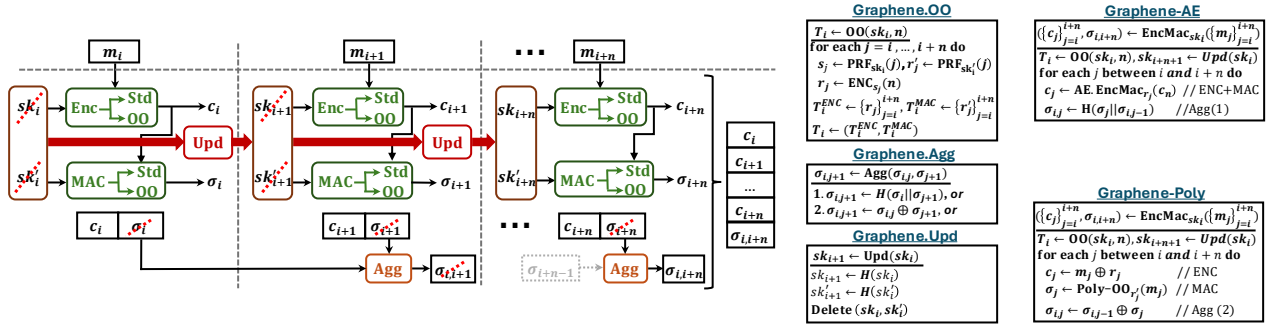
Fig. 3: Overview of `Graphene` Framework

Graphene.OO
$T_i \leftarrow \text{OO}(sk_i, n)$
for each $j = i, ..., i+n$ do
  $s_j \leftarrow \text{PRF}_{sk_i}(j), r'_j \leftarrow \text{PRF}_{sk'_i}(j)$
  $r_j \leftarrow \text{ENC}_{s_j}(n)$
$T_i^{ENC} \leftarrow \{r_j\}_{j=i}^{i+n}, T_i^{MAC} \leftarrow \{r'_j\}_{j=i}^{i+n}$
$T_i \leftarrow (T_i^{ENC}, T_i^{MAC})$

Graphene.Agg
$\sigma_{i,j+1} \leftarrow \text{Agg}(\sigma_{i,j}, \sigma_{j+1})$
1. $\sigma_{i,j+1} \leftarrow H(\sigma_{i,j} || \sigma_{j+1})$, or
2. $\sigma_{i,j+1} \leftarrow \sigma_{i,j} \oplus \sigma_{j+1}$

Graphene.Upd
$sk_{i+1} \leftarrow \text{Upd}(sk_i)$
$sk_{i+1} \leftarrow H(sk_i)$
$sk'_{i+1} \leftarrow H(sk'_i)$
Delete (sk_i, sk'_i)

Graphene-AE
$(\{c_j\}_{j=i}^{i+n}, \sigma_{i,i+n}) \leftarrow \text{EncMac}_{sk_i}(\{m_j\}_{j=i}^{i+n})$
$T_i \leftarrow \text{OO}(sk_i, n), sk_{i+n+1} \leftarrow \text{Upd}(sk_i)$
for each j between i and $i+n$ do
  $c_j \leftarrow \text{AE.EncMac}_{r_j}(c_n)$ // ENC+MAC
  $\sigma_{i,j} \leftarrow H(\sigma_j || \sigma_{j-1})$ //Agg(1)

Graphene-Poly
$(\{c_j\}_{j=i}^{i+n}, \sigma_{i,i+n}) \leftarrow \text{EncMac}_{sk_i}(\{m_j\}_{j=i}^{i+n})$
$T_i \leftarrow \text{OO}(sk_i, n), sk_{i+n+1} \leftarrow \text{Upd}(sk_i)$
for each j between i and $i+n$ do
  $c_j \leftarrow m_j \oplus r_j$ // ENC
  $\sigma_j \leftarrow \text{Poly-OO}_{r'_j}(m_j)$ // MAC
  $\sigma_{i,j} \leftarrow \sigma_{i,j-1} \oplus \sigma_j$ // Agg (2)

universal MAC schemes (e.g., LC [9]) support batch verification, which allows efficient tag verification. Otherwise, it generates individual tags and performs aggregation similar to `Graphene.EncMac` (Step 2-8).

B. Graphene Instantiations

Fig. 3 depicts a high-level overview of `Graphene` with its two main instantiations, described as follows:

Graphene-AE. It employs an AE with key update using a single private key. We use AES-GCM [7] (RFC 5288) that offers high efficiency due to hardware acceleration instructions on various platforms (e.g., AES-NI [8]). Moreover, GCM has OO with an online encryption time of only XOR operations. We use the first mode of aggregation with SHA-256. `Graphene-AE` provides: (1) High-level security with standard compliance (e.g., AES-GCM-{128,256} with SHA-{256,512}), (2) Computational efficiency even in the absence of OO with a single private key, (3) immutable aggregation.

Graphene-Poly. It uses AES-CTR for OO encryption and the universal MAC Poly1305 [6] (RFC 7539) for integrity. AES-CTR operates in CTR mode, while Poly1305 consists of a variant of universal MAC: $\text{Poly-OO}_k(m) = F_r(r, m) + s$ where $(r, s) \leftarrow \text{PRF}_k(n)$ is pre-computed offline. We use AES-128 as PRF and SHA-256 for intra-batch and inter-batch key updates, respectively. We adopt the second mode of aggregation (XOR) for optimal efficiency. `Graphene-Poly` provides: (1) faster running time compared to `Graphene-AE` while still having standard (RFC) compliance (2) optimal online computation for medium-to-standard ($\kappa = 103$-bit) security. Note that `Graphene-Poly` is extendable to other variants of Poly1305 MACs [6] with adjustable performance-security and up to 224-bit security but still without standard compliance.

C. Security Analysis

The security of `Graphene` relies on ENC and MAC schemes while considering forward security and tag aggregation:

Theorem 1. *If $\mathcal{A}$ can (t', ϵ', q_G)-break `Graphene`, then one can construct a PPT algorithm $\mathcal{B}$ to (t, ϵ, q_H)-break H, or (t, ϵ, q_P)-break F-IND-CPA-secure ENC, or (t, ϵ, q_S)-break FA-EU-CMA-secure MAC, respectively, where $t' = t + \mathcal{O}(q_H + q_P + q_S)$ and $\epsilon' = \epsilon \cdot i_0$ with i_0 being the maximum supported `Graphene.EncMac` operations.*

Proof. The F-IND-CPA security of ENC (instantiated with AES-CTR-128) is reduced to its underlying PRF (i.e., AES-128) and the pre-image resistance of the key update function H (i.e., SHA-256). $\mathcal{A}$ can $(t, \frac{1}{2^{129}}, q_P)$ and $(t, \frac{1}{2^{128}}, q_H)$-break AES-128 and SHA-256, respectively. The FA-EU-CMA of universal MACs is reduced to the security of universal hash functions (i.e., GHASH and Poly respectively), PRF, and H. $\mathcal{A}$ can $(t, \frac{1}{2^{128}}, q_S)$ and $(t, \frac{14 \cdot \frac{|m|}{16}}{2^{106}}, q_S)$-break GHASH [7] and Poly1305 [6], respectively, where $|m|$ is the input size.

`Graphene` provides fine-grained forward security by updating private keys per data item. Upon system compromise at $i < j' \leq i+n$, $\mathcal{A}$ do not have access to $\{sk_j\}_{j<j'}$ related to prior generated ciphertexts and tags. The only available keys to $\mathcal{A}$ are sk_{i+n+1} output of H, and the remaining one-time keys $\{r_j, r'_j\}_{j \geq j'}$ output of PRF_{sk_i} and $\text{PRF}_{sk'_i}$ in T_i. Thus, $\mathcal{A}$ decrypts the traffic associated with $\{sk_j\}_{j<j'}$ imply $(t, \frac{1}{2^{128}}, q_H)$-break H or $(t, \frac{1}{2^{129}}, q_P)$-break PRF. $\square$

Corollary 1. *$\mathcal{A}$ recovering individual authentication tags implies breaking the FA-EU-CMA-secure MAC.*

Proof. `Graphene` provides holistic integrity by outputting a sequential aggregate tags over a fixed batch size n: $\sigma_{i,i+n} \leftarrow$ `Graphene.Agg`$(\sigma_i, ..., \sigma_{i+n})$ where $\sigma_j \leftarrow \text{MAC}_{r'_j}(m_j)$, without revealing individual tags $\{\sigma_j\}_j$. Thus, any mix-and-match aggregate forgery [13] is detected if the queried batch of messages is not equal to the pre-selected window size n. $\square$

IV. PERFORMANCE EVALUATION

Our experimental configuration is as follows:
Commodity Hardware: We used a desktop equipped with an Intel i9-9900K@3.6 GHz processor and 64GB of RAM.
ARM Cortex M4: We used STM32F439ZI with 32-bit ARM Cortex-M4 CPU operating at 168 MHz and having 2MB flash memory and 256KB SRAM. It offers cryptographic acceleration (AES-{128,192,256}), SHA-256, and HMAC. Cortex-M4 is widely used in IoT due to its low cost and energy efficiency.
Software: We used OpenSSL[1] and GMP[2] to implement cryptographic primitives and arithmetic operations, respectively, on

---

[1] https://github.com/openssl/openssl
[2] https://gmplib.org

Scheme	Storage Overhead	Transmission Overhead	Computational Overhead			Security (κ)	FSec	OO	ImAgg	SC	BC
			Offline EncMac	Online EncMac	Online Verification						
Standard FAAE	2κ	$n \cdot |c| + 2\kappa$	$2n \cdot PRF + H$	$n \cdot (Enc + HMAC + H)$	$n \cdot (HMAC + H)$	128/256	✓	✗	✓	✓	✓
`Graphene-AE`	$\kappa + n \cdot |c|$	$n \cdot |c| + \kappa$	$n \cdot (PRF + AES.Enc_{|m|}) + H$	$n \cdot (GHASH + XOR_{|c|} + H)$	$n \cdot (GHASH + H)$	128/256	✓	✗	✓	✓	✓
`Graphene-Poly`	$2\kappa + n \cdot (\kappa + |c|)$	$n \cdot |c| + \kappa$	$n \cdot (PRF + AES.Enc_{|m|}) + H$	$n \cdot (Poly + XOR_{|c|} + XOR_{\kappa})$	$n \cdot (Poly + XOR_{\kappa})$	103/246	✓	✓	✗	✓	✓

κ denotes security level. n denotes the batch size. $|m|$ and $|c|$ denote the plaintext and ciphertext sizes. Add_q and Mul_q denote modular addition and multiplication over modulus q. Poly and GHASH denote one call to Poly1305 [6] and authenticator of AES-GCM [7], respectively. XOR_ℓ denotes XOR operations on ℓ-bit input. FSec, OO, ImAgg, SC, BC denote forward security, offline-online MAC capability, immutable aggregation, standard compliance, and backward compatibility, respectively.

TABLE I: Analytical storage and computational overhead of `Graphene` variants on batch of input messages

commodity hardware. We opt for wolfSSL[3] to implement both on Cortex-M4 due to its high efficiency and small code size.

Our evaluation metrics are: online computational overhead, tag size, breach resiliency, precomputation storage overhead, and extended properties (e.g., standard compliance).

A. Performance Analysis of Our Instantiations

We analyze the performance of `Graphene`, analytically and experimentally, as shown in Table I and Fig. 4.

We selected a standard FAAE scheme (e.g, [2]) as our baseline counterpart without precomputation. We used AES-CBC-128 and HMAC-SHA-256 as `ENC-Std` and `MAC-Std`, respectively. Both of aggregation and key update use SHA-256 as H. We consider the hardware acceleration of AES, SHA2, and HMAC on both platforms using AES-NI and Cortex optimizations. Our comparison shows the efficiency of UMACs with special properties (without hardware acceleration support) and the benefits of precomputation:

Graphene-AE. It uses AES-GCM with `OO`, whose online phase consists only of bitwise XOR operations for encryption but GHASH for authentication. As depicted in Fig. 4, on Cortex-M4 and for a batch of 1024 small 16-Byte inputs, the amortized running time is $2.55\times$ faster than standard FAAE, while online being $3.5\times$ faster with additional 16KB storage. On a batch of 1024 large 128-Byte inputs, the running times become $1.93\times$ and $3.5\times$ faster with 130KB extra storage, respectively. As for commodity, for a batch of 1024 small 16-Byte inputs, the amortized and online running times are $2.4\times$ and $4.3\times$ faster than that of FAAE. Overall, `Graphene-AE` offers efficient and scalable performance in offline and online phases while benefiting from hardware acceleration on both platforms.

Graphene-Poly. It uses AES-CTR-128 and Poly1305 as `ENC-OO` and `MAC-OO`, respectively. AES offers efficient encryption while Poly (RFC7539 standard) is ideal as `MAC-OO` with XOR as the choice of aggregation mode. Fig. 4 shows that `Graphene-Poly` achieves the best online running time on both platforms. For example, on Cortex-M4, its overall running time is $8.1\times$ faster than standard FAAE for a batch of 1024 16-Byte inputs, requiring additional storage of only 32KB. Its online running time is $9.23\times$ and $1.35\times$ faster than `Graphene-AE` on a batch of 1024 16-Byte and 1KB inputs, respectively, with only an additional 16KB. However, it incurs double storage compared to `Graphene-AE` (i.e., not a standalone AE). Overall, `Graphene-Poly` is the optimal

choice for medium-standard security ($\kappa = 106$-bit) with efficient online and overall running time.

Discussions. `Graphene-AE` is best suited for applications requiring high security with efficient online and total running times. `Graphene-Poly`, while incurring some extra storage, is the best choice for efficient online running time for medium-standard security. Moreover, `Graphene` can be easily extended with different MACs to provide additional properties (e.g., [4], [1]). For example, on small inputs (1-4 Bytes), BP-MAC [1] achieves the fastest online running time, outperforming Poly1305 by a factor of two on 2-Byte inputs. `Graphene` can also be combined with lightweight secret-sharing schemes for multipath communications [18] to recover and validate individual messages if aggregate verification fails, thus supporting fine-grained auditability.

V. CONCLUSION

We present `Graphene`, a symmetric-key-based FAAE framework designed for resource-constrained IoT systems, addressing critical limitations of existing MAC and AE schemes. By harnessing the synergy between forward-secure key evolution, offline-online cryptographic processing, and aggregate authentication, `Graphene` achieves breach resilience, near-optimal online latency, and compact tag generation. Our experimental evaluation on commodity hardware and 32-bit ARM Cortex-M4 microcontrollers demonstrates significant performance gains over existing FAAE solutions, with flexible instantiations to accommodate diverse security and efficiency requirements. We release a full-fledged open-source implementation to promote further research and real-world adoption.

REFERENCES

[1] E. Wagner, M. Serror, K. Wehrle, and M. Henze, "Bp-mac: fast authentication for short messages," in *15th ACM conf. on security and privacy in wireless and mobile networks*, pp. 201–206, 2022.

[2] A. A. Yavuz and P. Ning, "Self-sustaining, efficient and forward-secure cryptographic constructions for unattended wireless sensor networks," *Ad Hoc Networks*, vol. 10, no. 7, pp. 1204–1220, 2012.
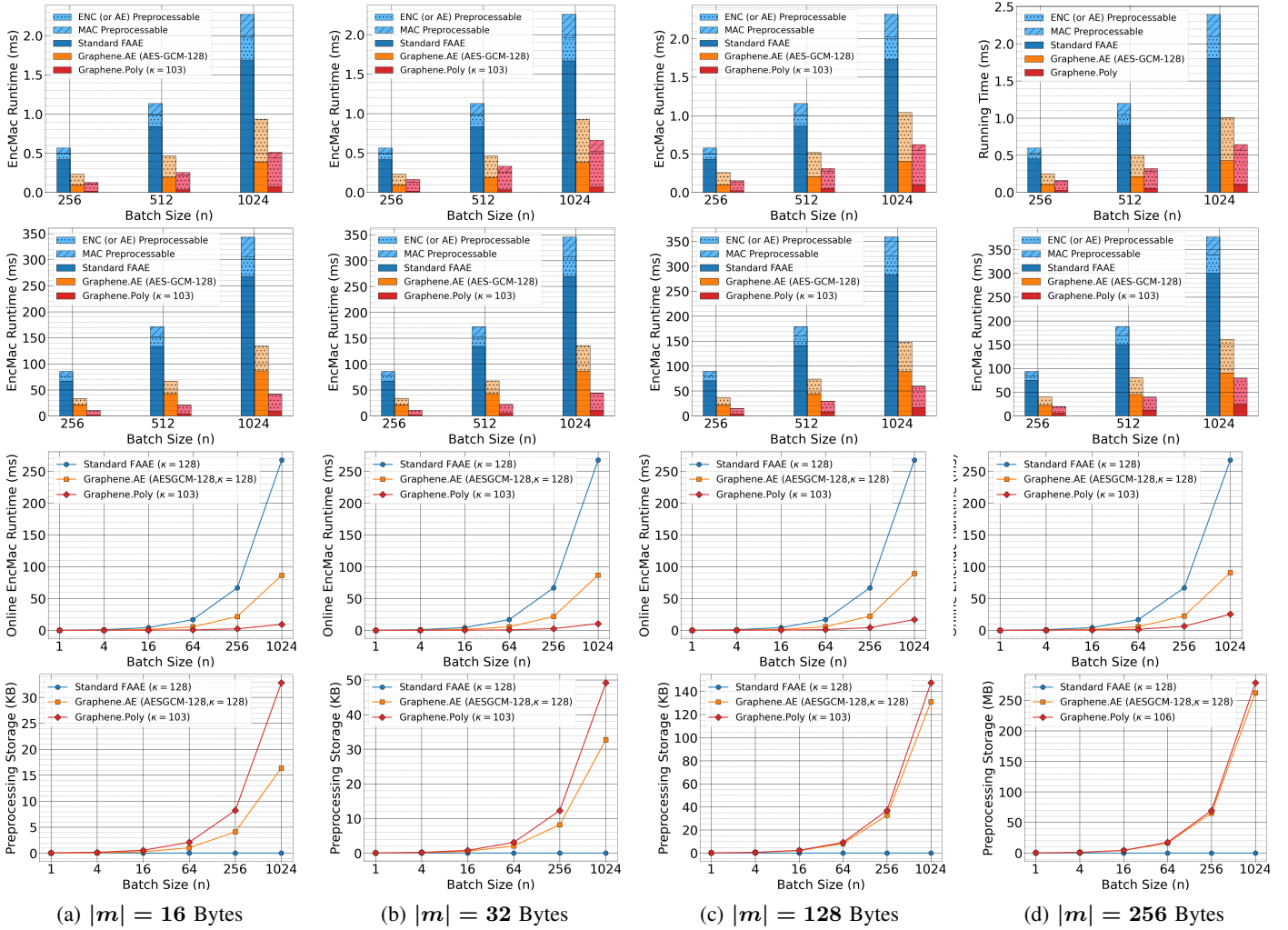
[3]https://github.com/wolfSSL/wolfssl/tree/master/IDE/STM32Cube

Fig. 4: OO running time and preprocessing storage overhead of `Graphene` variants on commodity hardware (x86/64: 1st row) and 32-bit ARM Cortex-M4 (2nd and 3rd rows)

[3] J. Hiller, M. Henze, M. Serror, E. Wagner, J. N. Richter, and K. Wehrle, "Secure low latency communication for constrained industrial iot scenarios," in *2018 IEEE 43rd Conf on Local Computer Networks*.

[4] E. Dubrova, M. Näslund, G. Selander, and F. Lindqvist, "Lightweight message authentication for constrained devices," in *Proc. of 11th ACM Conf. on Sec. & Priv. in Wireless and Mob. Nets.*, pp. 196–201, 2018.

[5] T. Yaqoob, H. Abbas, and M. Atiquzzaman, "Security vulnerabilities, attacks, countermeasures, and regulations of networked medical devices—a review," *IEEE Com. Surveys & Tutorials*, vol. 21, no. 4, 2019.

[6] J. P. Degabriele, J. Gilcher, J. Govinden, and K. G. Paterson, "Sok: Efficient design and implementation of polynomial hash functions over prime fields," in *IEEE Sym. on Security and Privacy (SP)*, 2024.

[7] D. A. McGrew and J. Viega, "The security and performance of the galois/counter mode (gcm) of operation," in *International Conference on Cryptology in India*, pp. 343–355, Springer, 2004.

[8] G. Hofemeier and R. Chesebrough, "Introduction to intel aes-ni and intel secure key instructions," *Intel, White Paper*, vol. 62, p. 6, 2012.

[9] M. Etzel, S. Patel, and Z. Ramzan, "Square hash: Fast message authentication via optimized universal hash functions," in *Annual International Cryptology Conference*, pp. 234–251, 1999.

[10] T. Le, P. Huang, A. A. Yavuz, E. Shi, and T. Hoang, "Efficient dynamic proof of retrievability for cold storage," *NDSS*, 2023.

[11] D. Ma and G. Tsudik, "Forward-secure sequential aggregate authentication," in *2007 IEEE Symposium on Security and Privacy*, May 2007.

[12] E. Wagner, M. Serror, K. Wehrle, and M. Henze, "When and how to

aggregate message authentication codes on lossy channels?," in *Int. Conf. on Applied Cryptography and Network Security*, pp. 241–264, 2024.

[13] O. Eikemeier, M. Fischlin, J.-F. Götzmann, A. Lehmann, D. Schröder, P. Schröder, and D. Wagner, "History-free aggregate message authentication codes," in *Security and Cryptography for Networks: 7th Int. Conf. SCN, September 13-15, 2010. Proceedings 7*, pp. 309–328.

[14] H. Li, L. He, S. Chen, J. Guo, and W. Qiu, "Automatic preimage attack framework on ascon using a linearize-and-guess approach," *IACR Transactions on Symmetric Cryptology*, 2023.

[15] E. Wagner, J. Bauer, and M. Henze, "Take a bite of the reality sandwich: revisiting the security of progressive message authentication codes," in *Proceedings of the 15th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, pp. 207–221, 2022.

[16] S. E. Nouma and A. A. Yavuz, "Trustworthy and efficient digital twins in post-quantum era with hybrid hardware-assisted signatures," *ACM Transactions on Multimedia Computing, Communications and Applications*, vol. 20, no. 6, pp. 1–30, 2024.

[17] S. E. Nouma and A. A. Yavuz, "Post-quantum forward-secure signatures with hardware-support for internet of things," in *ICC 2023-IEEE International Conference on Communications*, pp. 4540–4545, IEEE, 2023.

[18] A. Jha, S. Kashani, M. Hossein, A. Kirchner, M. Zhang, R. A. Chou, S. W. Kim, H. M. Kwon, V. Marojevic, and T. Kim, "Enhancing nextg wireless security: A lightweight secret sharing scheme with robust integrity check for military communications," in *IEEE Military Communications Conference (MILCOM)*, pp. 1–6.