

Article

Lightweight One-to-Many User-to-Sensors Authentication and Key Agreement

Hussein El Ghor ^{1,2} , Ahmad Hani El Fawal ^{1,3} , Ali Mansour ³ , Ahmad Ahmad-Kassem ⁴
and Abbass Nasser ^{5,*} 

¹ ARCAS Research Center, Computer Science Department, Modern University for Business and Science, Beirut P.O. Box 113-7501, Lebanon; aelfawal@mubs.edu.lb (A.H.E.F.)

² Faculty of Technology, Lebanese University, Saida P.O. Box 6573/14, Lebanon

³ Lab-STICC, UMR 6285—CNRS, ENSTA, Institut Polytechnique de Paris, 29200 Brest, France

⁴ Faculty of Arts & Sciences, Holy Spirit University of Kaslik, Jounieh P.O. Box 446, Lebanon

⁵ CIRAME Research Center, Business School, Holy Spirit University of Kaslik, Jounieh P.O. Box 446, Lebanon

* Correspondence: abbass.nasser@usek.edu.lb

Abstract

The proliferation of Internet of Things (IoT) deployments demands Authentication and Key Agreement (AKA) protocols that scale from one initiator to many devices while preserving strong security guarantees on constrained hardware. Prior lightweight one-to-many designs often rely on a network-wide secret, reuse a single group session key across devices, or omit Perfect Forward Secrecy (PFS), leaving systems vulnerable to compromise and traffic exposure. To this end, we present in this paper a lightweight protocol, named Lightweight One-To-many User-to-Sensors Authentication and Key Agreement (LOTUS-AKA), that achieves mutual authentication, PFS, and per-sensor key isolation while keeping devices free of public-key costs. The user and gateway perform an ephemeral elliptic-curve Diffie–Hellman exchange to derive a short-lived group key, from which independent per-sensor session keys are expanded via Hashed Message Authentication Code HMAC-based Key Derivation Function (HKDF). Each sensor receives its key through a compact Authenticated Encryption with associated data (AEAD) wrap under its long-term secret; sensors perform only hashing and AEAD, with no elliptic-curve operations. The login path uses an augmented Password-Authenticated Key Exchange (PAKE) to eliminate offline password guessing in the smart-card theft setting, and a stateless cookie gates expensive work to mitigate denial-of-service. We provide a game-based security argument and a symbolic verification model, and we report microbenchmarks on Cortex-M-class platforms showing reduced device computation and linear low-constant communication overhead with the number of sensors. The design offers a practical path to secure, scalable multi-sensor sessions in resource-constrained IoT.

Keywords: IoT security; authentication; key agreement; elliptic-curve cryptography; perfect forward secrecy; PAKE; AEAD; batch authentication



Academic Editors: Yang-Wai Chow,
Nan Li and Chau Nguyen

Received: 29 November 2025

Revised: 16 December 2025

Accepted: 23 December 2025

Published: 4 January 2026

Copyright: © 2026 by the authors.
Licensee MDPI, Basel, Switzerland.
This article is an open access article
distributed under the terms and
conditions of the [Creative Commons
Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

The development of Internet of Things (IoT) technology has greatly simplified people's lives, allowing them to access real-time data from sensors thousands of miles away without leaving their homes [1]. However, due to the inherent nature of data transmission over public channels, incidents of information interception and tampering are a common occurrence [2].

Large-scale implementations of IoT regularly require an initiator (a user or a gateway) to establish protected channels with multiple sensors [3]. Repeating a one-to-one Authentication and Key Agreement (AKA) handshake n times wastes spectrum and energy, inflates latency, and magnifies the exposed attack surface at constrained devices [4]. The reason is two-fold: (i) each node must expose and parse additional handshake/certificate code paths that empirical studies show, which are prone to state-machine and parsing flaws exploitable for DoS or logic errors [5–7], and (ii) deployments of handshake-gating mechanisms (e.g., address-validation/anti-amplification) can still be bypassed or abused, leaving residual flooding vectors [8–10].

These pressures motivate one-to-many AKA: a single protocol run that yields fresh keys with a set of sensors while keeping device-side costs low. Prior “lightweight” approaches push public-key work to a more capable party and leave hashing or simple symmetric operations at sensors, achieving practical performance in IoT and Industrial IoT settings [11–15]. However, many such designs remain either (i) heavy on devices under realistic batching and revocation demands, or (ii) miss core guarantees such as Perfect Forward Secrecy (PFS), per-sensor key isolation, and a concrete rekey path, which are essential in adversarial, high-churn deployments [16].

1.1. Limitations of Prior Scheme(s)

Lack of true PFS: Several one-to-many schemes derive final session material primarily from long-term server, device secrets or static group values, so post-compromise disclosure of these values can retrospectively expose past traffic. In particular, designs that omit an ephemeral Diffie–Hellman (DH) contribution in the key schedule cannot claim PFS.

To overcome this limitation, we confine a fresh ephemeral Elliptic Curve Diffie–Hellman (ECDH) to the user–gateway path and base the entire session on that secret.

Single network-wide secret: Some constructions rely on a gateway-chosen value shared across all sensors (e.g., to mask identities or authenticate broadcasts). Compromising any one node then deanonymizes or weakens the entire fleet.

In this paper, we eliminate network-wide secrets; all derivations and authenticity checks are conducted under per-sensor long-term keys managed at the gateway registry.

Group key shared by all sensors: To save messages, a single group session key is often delivered to every sensor in the batch. This couples confidentiality across devices: the weakest sensor compromises the rest.

To fix this weakness, we adopt a two-tier schedule—derive a short-lived group key, then expand per-sensor keys—to deliver each securely via a compact Authenticated Encryption with associated data (AEAD) wrap under that sensor’s long-term secret.

Offline guessing under smart-card theft: Login paths that store verifier material on the card (or server) without augmentation enable offline dictionary attacks if the card or database is copied.

To mitigate this attack, we integrate an augmented Password-Authenticated Key Exchange (PAKE) so the server-side secret is required to validate guesses, hence, closing the stolen-card gap.

Timestamp-only DoS control: Many “lightweight” designs accept messages if a timestamp falls within a window, which is cheap for attackers to satisfy and does not gate expensive operations.

This limitation is fixed using a stateless cookie/Message Authentication Code (MAC) before-work gate at the gateway with minimal checks at sensors so unauthenticated requests cannot trigger heavyweight processing.

1.2. Our Contributions

In this paper, we present a Lightweight One-To-many User-to-Sensors Authentication and Key Agreement (LOTUS-AKA) protocol that preserves device frugality while restoring missing guarantees.

1. One-to-many with PFS, no Elliptic-Curve Cryptography (ECC) on sensors: a single ephemeral ECDH between user and gateway yields a secret session; sensors perform only hashing and AEAD, without any elliptic-curve operations.
2. Per-sensor keys without network-wide secrets: we derive a short-lived group key GK, expand per-sensor, and deliver them via AEAD wraps under each sensor's long-term key, thus removing any shared masking value.
3. Augmented-PAKE login: the user login is hardened with an augmented PAKE, eliminating offline password guessing even if a smart card or server database is copied.
4. Stateless DoS cookie: a MACed cookie authenticates the initiator before any heavy-weight work at the gateway; sensors verify a cheap tag before state allocation.
5. Revocation and rekey: we provide a Logical Key Hierarchy (LKH)-style mechanism with subset wrapping to revoke sensors and rotate long-term keys with bounded broadcast cost.
6. Security analysis: we provide an Extended Canetti-Krawczyk eCK/CK⁺-style proof for AKE security and privacy (anonymity/unlinkability) [17] and validate the design with a ProVerif/Tamarin model [18].
7. Performance: measurements on Cortex-M-class Microcontroller Units (MCUs) show lower device computation and linear, low-constant communication/energy versus repeated one-to-one handshakes and recent one-to-many baselines.

The rest of the paper is organized as follows: Previous works is presented and discussed in Section 2. Section 3 introduces the background and the used preliminaries. The LOTUS-AKA Protocol is proposed in Section 4. In Section 5, we present the complete security analysis for the LOTUS-AKA Protocol. The evaluation of the proposed protocol is performed in Section 6. Section 7 presents the conclusion with future works.

2. Related Work

This section reviews prior work most relevant to lightweight, one-to-many AKA for IoT and WSN deployments. We group the literature into (i) standard one-to-one AKEs for constrained devices; (ii) broadcast and group-oriented authentication; (iii) group key management and subset rekeying; (iv) smart-card and multi-server schemes adapted to IoT; (v) DoS and availability defenses in handshakes; and (vi) symmetric-crypto choices and KDFs on MCUs.

2.1. One-to-One AKE for Constrained Devices

Transport-layer designs such as Datagram Transport Layer Security (DTLS) deliver mutual authentication and channel security over datagrams; DTLS 1.2 introduced a stateless cookie to reduce amplification, while DTLS 1.3 streamlined the handshake along TLS 1.3 lines [19,20]. Application-layer protection via OSCORE moves security to the Constrained Application Protocol (CoAP) message layer and composes cleanly with constrained transports [21]. Ephemeral Diffie-Hellman Over COSE (EDHOC), standardized by the Lightweight Authenticated Key Exchange Working Group (IETF LAKE WG), offers a compact AKE that pairs naturally with OSCORE [22–24]. These protocols are mature and well analyzed, but they are fundamentally one-to-one: a controller addressing n sensors typically repeats the handshake n times ($1 \rightarrow 1$ repeated), amplifying ECDH/PAKE cost and airtime. Moreover, if sensors participate in ECC, the device code, energy, and timing budget grow accordingly.

2.2. Broadcast and Group-Oriented Authentication

Early WSN systems pushed cost to the sender and the network using symmetric pre-distribution and delayed-disclosure broadcast authentication (TESLA and variants) to avoid per-packet public-key work [25]. Later controller-to-many proposals derive a group key once and distribute per-node material. However, many “lightweight” designs optimize device code by (i) assuming a network-wide secret shared across verifiers, (ii) masking a batch group key with hashes or XOR without binding to a full transcript, and (iii) relying on timestamps for freshness. As a result, compromise of a single node or the controller’s long-term secret can retroactively expose past sessions, and simple cut-and-paste across runs may succeed when identities and context are not authenticated in associated data. In contrast, our design keeps the sender convenience (one broadcast plus n short ACKs) but eliminates network-wide secrets, derives a Group Key GK from an ephemeral ECDH visible only to users and gateways, and AEAD-wraps per-sensor shares under sensor keys with transcript-bound associated data (AD), achieving per-sensor isolation and modern freshness.

2.3. Group Key Management and Subset Rekeying

Logical Key Hierarchy (LKH) and related key-graph methods achieve scalable rekeying for a number of group members n (users/devices) in the secure group: joins and leaves cost $O(\log n)$ messages instead of $O(n)$ by organizing members in a tree [26,27].

Subset-cover broadcast encryption (e.g., the subset-difference construction) provides alternative trade-offs among cover size, state, and revocation latency [28]. These tools power secure multicast, pay-TV, and IoT fleet management. Our protocol composes naturally with either approach: the gateway can wrap GK to the addressed subset using per-sensor AEADs (our default) or compact subset covers without changing device code. Optional key evolution on devices further limits damage from late device compromise when secure erasure is feasible.

2.4. Smart-Card and Multi-Server Authentication in IoT

A large thread of the IoT literature adapted multi-server, smart-card-based logins to gateways and sensors. Despite being labeled “lightweight,” many such designs avoid public-key cryptography on devices using system-wide symmetric verifiers and timestamp-only replay control, or by masking batch keys with unbound hashes. Analyses repeatedly identified absence of true PFS (no ephemeral DH); offline dictionary attacks once a card is copied; cross-device compromise due to network-wide secrets; and transcript splicing because identities and freshness are not authenticated in an AEAD AD. Our approach addresses the exact issues enumerated in Section 1.1: the only public-key work is an ephemeral ECDH between U and G ; sensors do symmetric crypto only; user login is via augmented PAKE (OPAQUE-style), which eliminates offline guessing under card theft; and all ciphertexts are bound to identities and transcript via AEAD AD.

2.5. DoS and Availability in Handshakes

Stateless cookies gate expensive processing by requiring an initiator to echo a MACed token before the responder allocates state or runs heavy cryptography. This pattern appears in DTLS (1.2 HelloVerify, 1.3 cookie-based retry), Internet Key Exchange Protocol Version 2 (IKEv2) cookies, and Quick UDP Internet Connections (QUIC) retry with anti-amplification [19,20,29,30]. On constrained gateways and sensors, MAC-before-work, constant-time parsing, and bounded replay windows (timestamps plus counters) are critical to maintain availability under floods. Our cookie and pseudonym checks enforce these gates before any AEAD open or HKDF work.

2.6. Primitives on MCUs: AEAD and KDF

AEAD protects confidentiality and integrity while binding additional headers. AES-GCM is widely hardware-accelerated (ChaCha20–Poly1305 performs well in software on MCUs without AES engines [31,32]). For key derivation, KDF’s extract-then-expand design is simple and robust, with strong PRF-based proofs and broad deployment [33]. For ECDH between U and G , X25519 provides compact, constant-time scalar multiplication with ubiquitous library support [34]. Our evaluation echoes prior observations: on Cortex-M0, ChaCha20–Poly1305 minimizes per-sensor latency; on Cortex-M4/M7 with AES, AES-GCM is preferable.

2.7. Positioning

Table 1 presents a comparison among the existing protocols/families and our proposed LOTUS-AKA protocol with respect to several criteria. Standard $1 \rightarrow 1$ AKEs (DTLS, EDHOC+OSCORE) are excellent for pairwise channels but scale poorly when a user must reach many sensors at once. WSN-era broadcast techniques amortize sender cost but typically lack modern guarantees such as per-sensor isolation and PFS under device compromise. Smart-card multi-server proposals often fall to the Section 1.1 issues: network-wide keys, no ephemeral DH, timestamp-only replay control, and offline dictionary under card theft. LOTUS-AKA occupies the intersection: one ECDH per run (off-device), per-sensor AEAD wraps with transcript-bound AD, augmented PAKE login, stateless DoS cookie gates, and straightforward composition with LKH or subset covers.

Table 1. Summary comparison: prior families vs. LOTUS-AKA. Y = supported; N = not; $\sim$ = partial/depends. Communication scaling is total control-plane bytes vs. the number of sensors n .

Protocol/Family	MutAuth	PFS	Per-Sens Iso	KCI	Privacy	OffDict	DoS	ECC@Sens	Comm. vs. n	Revoc
DTLS 1.3 ($1 \rightarrow 1$) [20]	Y	Y	–	Y	$\sim$	Y (no card)	Y	optional	$O(n)$ (repeat)	–
EDHOC+OSCORE ($1 \rightarrow 1$) [22,24]	Y	Y	–	Y	$\sim$	Y (no card)	via CoAP	optional	$O(n)$ (repeat)	–
TESLA-style broadcast [25]	auth. only	$\sim$	N	–	N	–	N	Y (none)	$O(1)$ sender meta	$\sim$
Group key w/masking (typical)	Y	N	N	N	N	N	N	Y (none)	$O(1) + O(n)$ wraps	$\sim$
Smart-card multi-server (typical)	Y	N	N	N	$\sim$	N	N	Y (none)	$O(1) + O(n)$	$\sim$
LKH subset rekey [26–28]	–	$\sim$	$\sim$	–	N	–	N	Y (none)	$O(\log n)$ rekey	Y
LOTUS-AKA (this work)	Y	Y	Y	Y	Y	Y	Y	Y (none)	$O(1) + O(n)$	Y (via LKH)

3. Background and Preliminaries

3.1. Notation and Cryptographic Primitives

We use standard elliptic-curve groups over either NIST P-256 or X25519 for ephemeral Diffie–Hellman. Let G denote the base point, and let $x \in \mathbb{Z}_q$ be a uniformly random scalar with public point $X = xG$. Key derivation employs HKDF with $H = \text{SHA256}$; message authentication uses HMAC over H . Authenticated Encryption uses an AEAD scheme (either AES–GCM or ChaCha20–Poly1305) with associated data (AD) binding identities, counters, timestamps, and transcript hashes. The password login is instantiated with an augmented PAKE with OPAQUE-style, so the server stores a hardened record while the client holds only an opaque envelope (for complete Notation summary, see Table 2).

Security Assumptions and Proof Oracles

Our game-based analysis relies on hardness of CDH/X25519, PRF/RO behavior of H and HKDF, and IND–CCA security of the AEAD. We use standard AKE oracles: Send (activate a protocol step), Reveal (expose a completed session key), EphemeralReveal (expose fresh DH scalars in eCK/CK^+), StateReveal (long-term state), Corrupt (compromise of users/sensors), and Test (real-or-random key indistinguishability). Privacy is modeled via anonymity/unlinkability queries that challenge the adversary to distinguish identities or link sessions given full transcripts and allowable side information. Freshness excludes

sessions where the attacker learns the joint DH secret or both parties' long-term keys prior to Test.

Table 2. Notation summary.

Symbol	Meaning
G	Base point of P-256/X25519 group
x, X	Private scalar $x \leftarrow \mathbb{Z}_q$, public $X = xG$
s	Gateway master secret
SID_j	Identifier of sensor S_j
sk_j	Long-term per-sensor secret $H(SID_j \parallel s)$
Y	Gateway ephemeral yG ; X is user ephemeral
Z	ECDH secret xyG ; $GK = HKDF(Z, \text{ctx})$
SK_j	Per-sensor session key $HKDF(GK, SID_j \parallel \text{transcript})$
π_j	Pseudonym $HMAC_{sk_j}(TS \parallel \text{ctr})$
Wrap_j	$AEAD_{sk_j}(GK; AD)$ to deliver SK_j
TS, ctr	Timestamp and per-session counter
$H, HKDF$	SHA-256 hash and HKDF (extract&expand)
$AEAD$	AES-GCM or ChaCha20-Poly1305

3.2. System and Trust Model

Before we start with the system and trust model, it is necessary to define some conventions and notations:

All protocol values are byte strings. We write $\parallel$ for bitwise concatenation. A deterministic, injective encoder $\text{enc}(\cdot)$ serializes structured values to bytes: $\text{enc}(SID_j)$ is the length-prefixed UTF-8 encoding of the sensor identifier; $\text{enc}(s)$ is the fixed-length big-endian encoding of the κ -bit secret s .

λ : Output length of the hash (bits), e.g., $\lambda = 256$.

$H : \{0, 1\}^* \rightarrow \{0, 1\}^\lambda$: Cryptographic hash (e.g., SHA-256).

$SID_j \in \{0, 1\}^*$: Public identifier of sensor S_j .

$s \in \{0, 1\}^\kappa$: Gateway master secret.

$\parallel$: Byte-string concatenation.

$\text{enc}(\cdot)$: Deterministic, injective byte encoding.

Entities: A user U (operator/app), a gateway G (registry, policy, and broadcast engine), and a set of sensors $\{S_j\}$ (resource constrained, symmetric-only).

Keys: The gateway samples a master secret $s \in \mathbb{Z}_q$. For each sensor with identifier SID_j , the long-term per-sensor secret is presented as follows:

$$sk_j = H(SID_j \parallel s).$$

$$\begin{aligned} sk_j &= H(SID_j \parallel s) \\ &= H(\text{enc}(SID_j) \parallel \text{enc}(s)). \end{aligned}$$

The gateway maintains a registry $\langle SID_j, sk_j \rangle$. Users authenticate with G via an augmented PAKE credential; no plaintext password verifiers are stored at G , and user devices hold only an opaque PAKE envelope.

Adversary: We assume a Dolev–Yao network adversary with full control over delivery and ordering of messages (eavesdrop, inject, replay, and delay). The adversary may attempt device capture (including extraction of a smart card or a sensor), offline dictionary attacks, and partial compromise of server-side state. Clocks are loosely synchronized with a skew

bound Δ ; protocols additionally bind nonces/counters to each session to prevent replays even under clock drift.

We model loose synchronization by a bound Δ such that $\forall i, j, t: |C_i(t) - C_j(t)| \leq \Delta$. In practice set $\Delta = 2\rho_{\max}\tau + \varepsilon_{\text{sync}}$ from oscillator drift $\rho_{\max}$, resync interval τ , and residual offset $\varepsilon_{\text{sync}}$, where:

- $C_i(t)$: Local clock reading of node i at physical time t (seconds), assumed monotone.
- $\rho_{\max}$: Worst-case relative drift (frequency error) of clocks (e.g., in ppm).
- τ : Upper bound on the interval between (re-)synchronization events.
- $\varepsilon_{\text{sync}}$: Residual clock offset immediately after a sync exchange (accounts for delay asymmetry).

3.3. Design Goals

Our protocol targets the following properties:

- Mutual authentication between U and each S_j via G .
- Perfect Forward Secrecy (PFS) from an ephemeral ECDH between U and G .
- Key-compromise impersonation (KCI) resistance and key indistinguishability.
- Privacy: user anonymity and sensor unlinkability from public transcripts.
- Per-sensor confidentiality: even within one batch run, keys SK_j are isolated across sensors.
- Replay and DoS resistance: timestamps + nonces/counters, and a stateless MACed cookie that gates expensive work.
- Efficiency and scalability: sensors perform only hash/HMAC/AEAD; communication scales linearly with the number of addressed sensors with small constants.
- Revocation/rekey: operator can revoke sensors and rotate long-term material using a lightweight Logical Key Hierarchy (LKH) with subset wrapping.

4. Lightweight One-to-Many User-to-Sensors Authentication and Key Agreement (LOTUS-AKA) Protocol

4.1. Setup and Registration

Algorithm 1 outlines the setup and registration part. In details:

Gateway: The gateway G selects an elliptic-curve group (P-256 or X25519) with base point P , and samples a master secret $s \leftarrow \mathbb{Z}_q$.

Sensor registration: For each sensor with identifier SID_j , G derives a long-term per-sensor secret

$$sk_j = H(SID_j \parallel s),$$

and stores the registry entries $\langle SID_j, sk_j \rangle$. No network-wide key is ever created or distributed. User registration/login: Users authenticate to G via an augmented PAKE with OPAQUE-style. The server stores a hardened PAKE record; the user device (or card) holds only an opaque envelope. Thus, neither side alone enables offline password guessing if compromised. In other words, we store a hardened PAKE record as a verifier, not the password. That means, instead of storing the password, the server stores a hardened verifier produced with a memory-hard password hashing function (Argon2id) using a per-user salt and tuned cost parameters, optionally protected with a server-side pepper/HSM [35,36]. With OPAQUE, the server keeps only an OPRF seed and an encrypted credential “envelope” created at registration; the server never learns the password and a leaked database does not enable offline guessing [37,38]

Algorithm 1: LOTUS-AKA SETUP AND REGISTRATION.

State: Gateway registry $\mathcal{R}$ initially empty

- 1 **Gateway G:** choose curve (P-256 or X25519) with base point P ; sample master secret $s \leftarrow \mathbb{Z}_q$;
- 2 **Sensor registration (each S_j with identifier SID_j):**
- 3 $sk_j \leftarrow H(SID_j \parallel s)$; store $\langle SID_j, sk_j \rangle$ in $\mathcal{R}$ (no network-wide key);
- 4 **User registration/login (augmented PAKE, OPAQUE-style);**
- 5 server stores a hardened PAKE record; user device/card stores an opaque envelope (no plaintext password);

Sensors do not derive the current group key GK from their long-term secret. Instead, after the user–gateway run establishes a new GK , the gateway distributes GK to each addressed sensor S_j by encapsulating it with Authenticated Encryption under the sensor’s secret key sk_j . Concretely, the gateway includes in the broadcast a per-sensor identifier π_j and an AEAD ciphertext $Wrap_j$ that encrypts GK with associated data binding (e.g., SID_j , timestamp/nonce, and epoch). Each sensor computes its own π_j , locates the matching tuple in the broadcast, and performs $AEAD_Open_{sk_j}(Wrap_j)$ to recover GK , from which it derives its per-sensor session key SK_j using the specified key-derivation step.

By default, GK is refreshed on every one-to-many authentication session because it is derived from fresh user–gateway ephemeral material.

4.2. One-to-Many Authentication and Key Agreement

Let $\mathcal{S} = \{j_1, \dots, j_m\}$ be the subset of addressed sensors in this run, $b \leftarrow \mathbb{Z}_q$ the user’s fresh scalar with $X = bP$, and $y \leftarrow \mathbb{Z}_q$ the gateway’s fresh scalar with $Y = yP$. Let τ denote the running transcript hash bound into all KDFs/AEAD AD. Figure 1 illustrates a concrete example of subset-tag encoding and gateway-side expansion.

Algorithm 2 tracks the steps to apply the one-to-many authentication and key agreement.

(1) User $\rightarrow$ Gateway

The user U sends a request containing (i) the PAKE finish tag, (ii) a response to a stateless DoS cookie issued earlier by G , (iii) the sensor subset descriptor for $\mathcal{S}$ (optionally anonymized as a policy tag), and (iv) the ephemeral $X = bP$. Sensor identifiers do need not to appear clearly when anonymity is required.

(2) Gateway processing

Upon verifying the PAKE finish and cookie, G samples y , sets $Y = yP$, and computes the ECDH shared secret $Z = KDF(bY) = KDF(yX)$. It derives a short-lived group key

$$GK = HKDF(Z, \text{info} = \tau),$$

which never leaves U and G clearly.

τ is the HKDF info context string providing domain separation.

(3) Per-sensor wrapping and broadcast

For each $j \in \mathcal{S}$, G computes a per-run pseudonym and wrap:

$$\begin{aligned} \pi_j &= \text{HMAC}_{sk_j}(\text{TS} \parallel \text{ctr}), \\ \text{Wrap}_j &= \text{AEAD}_{sk_j}(GK; \text{AD} = \text{SID}_j \parallel \text{TS} \parallel \tau). \end{aligned} \tag{1}$$

G broadcasts a single message

$$\{Y, TS, ctr, \{\pi_j, \text{Wrap}_j\}_{j \in S}\},$$

optionally alongside a compact policy tag instead of explicit SID_j values (for anonymity; sensors recover by trial on π_j).

Algorithm 2: Unified One-to-Many Execution of the Proposed Protocol (LOTUS-AKA Run).

Input: Subset of addressed sensors S ; server-issued Cookie; aPAKE parameters
Output: Per-sensor session keys $\{SK_j\}_{j \in S}$ shared between U and each S_j
State: At G : registry $\mathcal{R}$ with $\langle SID_j, sk_j \rangle$; short-lived cookie key k_{cookie} ; master s (enrollment only).
At U : password for aPAKE; (optional) secure element/card with opaque envelope.
At S_j : long-term key sk_j ; local replay window.

- 1 **Constants and notation:** base point P ; fresh $b, y \in \mathbb{Z}_q$ with $X = bP$, $Y = yP$; transcript hash t ; KDF, HKDF, HMAC, AEAD; timestamp TS ; counter ctr ; associated data $AD = SID_j \parallel TS \parallel t$;
- 2 **(1) User initialization at U :**
- 3 Run augmented-PAKE with G to completion; compute the PAKE *finish* authenticator;
- 4 Verify and echo Cookie (stateless MAC-before-work);
- 5 Sample $b \leftarrow \mathbb{Z}_q$; set $X \leftarrow bP$;
- 6 Construct subset descriptor S (policy tag or explicit indices); avoid sending SID_j in clear if anonymity is required;
- 7 Send $\langle \text{PAKE_finish}, \text{Cookie_resp}, S, X \rangle$ to G ;
- 8 **(2) Gateway processing and single broadcast at G :**
- 9 Verify PAKE finish and Cookie; **if either fails then**
- 10 **return** abort
- 11 ;
- 12 Sample $y \leftarrow \mathbb{Z}_q$; set $Y \leftarrow yP$;
- 13 Compute $Z \leftarrow \text{KDF}(yX)$; update transcript hash $t \leftarrow H(\text{transcript} \parallel X \parallel Y)$;
- 14 Derive $GK \leftarrow \text{HKDF}(\text{IKM} = Z, \text{salt} = t, \text{info} = \text{"GK"})$;
- 15 Choose fresh (TS, ctr) ;
- 16 **foreach** $j \in S$ **do**
- 17 fetch sk_j ;
- 18 $p_j \leftarrow \text{HMAC}_{sk_j}(TS \parallel ctr)$; // per-run pseudonym
- 19 $AD_j \leftarrow SID_j \parallel TS \parallel t$;
- 20 $\text{Wrap}_j \leftarrow \text{AEAD}_{sk_j}(\text{plaintext} = GK; AD = AD_j)$;
- 21 Broadcast $\mathcal{B} = \{Y, TS, ctr, (p_j, \text{Wrap}_j)_{j \in S}\}$ to sensors;
- 22 Optionally cache (TS, ctr, t) until confirmations; discard on timeout;
- 23 **(3) Sensor processing at each S_j :**
- 24 Constant-time precheck; reject if (TS, ctr) outside window (drop early);
- 25 Compute $p^* \leftarrow \text{HMAC}_{sk_j}(TS \parallel ctr)$; scan $(p_\ell, \text{Wrap}_\ell)$ for $p_\ell \stackrel{?}{=} p^*$;
- 26 **if no match then**
- 27 **return** drop silently
- 28 ;

Algorithm 2: Cont.

```

29 Set  $AD_j \leftarrow SID_j \parallel TS \parallel t$ ;
30 Open  $GK \leftarrow AEAD\_open_{sk_j}(Wrap_\ell; AD_j)$ ; if fail then
31   | return abort
32 ;
33 Derive  $SK_j \leftarrow HKDF(IKM = GK, salt = t, info = SID_j \parallel UID \parallel t)$ ;
34 Compute confirmation  $MAC_j \leftarrow HMAC_{SK_j}(t)$ ; send  $MAC_j$  to  $G$ ;

35 (4) Gateway aggregation (optional) and relay;
36 Collect  $MAC_j$  from addressed sensors within a timeout; deduplicate retransmits;
37 Optionally relay  $\{Y, \{MAC_j\}_{j \in S}\}$  to  $U$  (or  $U$  observes directly);

38 (5) User finalization at  $U$ ;
39 Upon receiving  $Y$  and  $\{MAC_j\}$ , recompute  $Z \leftarrow KDF(bY)$  and refresh  $t$  consistently with  $G$ ;
40 Derive  $GK \leftarrow HKDF(Z, t, "GK")$ ;
41 foreach  $j \in S$  do
42   |  $SK_j \leftarrow HKDF(GK, t, SID_j \parallel UID \parallel t)$ ;
43   | Verify  $MAC_j \stackrel{?}{=} HMAC_{SK_j}(t)$ ; if any fails then
44     | return abort
45   | ;
46 Optional send final acknowledgement to  $G$ ; commit  $\{SK_j\}$ ;

47 (6) Freshness and DoS gates (all receivers);
48 Enforce MAC-before-work via Cookie; reject invalid/stale tokens before any AEAD/HKDF;
49 Bound replays with a sliding window over  $(TS, ctr)$ ; drop duplicates early;
50 Parsing/MAC checks are constant-time; allocate state only after gates pass;

51 (7) Optional forward-secure wraps (device key evolution);
52 If enabled after success:  $sk_j^{(e+1)} \leftarrow H(sk_j^{(e)})$  at  $S_j$  and in  $\mathcal{R}$ ; securely erase  $sk_j^{(e)}$ ;

```

Illustrative Example (subset tag / anonymized subset descriptor). Assume the user is authorized to access the sensor group named BuildingA/Floor3/Temp. Rather than sending explicit sensor identifiers on the user $\rightarrow$ gateway link, the user sends a fixed-length subset tag that represents this authorized policy.

Tag computation: Let the gateway maintain a per-policy secret k_{pol} shared with authorized users for that policy (distributed during enrollment/authorization). For epoch e (e.g., daily or per-session), the user computes

$$tag = Trunc_\ell(HMAC_{k_{pol}}("BuildingA/Floor3/Temp" \parallel e)),$$

where $Trunc_\ell(\cdot)$ keeps ℓ bits (e.g., $\ell=64$ or 128) to obtain a compact identifier.

Gateway expansion: Upon receiving tag, the gateway looks up tag in its local registry/ACL and expands it to the addressed subset $S = \{S_7, S_{19}, S_{24}, S_{31}\}$ (as example), without revealing these identifiers on the user $\rightarrow$ gateway link.

Unchanged sensor processing: The gateway then broadcasts the usual one-to-many message carrying per-sensor tuples $(\pi_j, Wrap_j)_{j \in S}$. Each sensor S_j computes its pseudonym π_j , finds the matching tuple, and opens $Wrap_j$ to recover GK and derive SK_j exactly as in the main LOTUS-AKA flow.

Figure 1. Concrete example of subset-tag encoding and gateway-side expansion.

(4) Sensor S_j processing

Each sensor verifies the cheap cookie/tag and finds its entry by testing π_j under sk_j . It opens $Wrap_j$ using AEAD to obtain GK , and derives its independent session key

$$SK_j = HKDF(GK, info = SID_j \parallel UID \parallel \tau).$$

It returns an authenticated confirmation $MAC_j = HMAC_{SK_j}(\tau \parallel \text{role})$ to G .

(5) User key confirmation

The user recomputes $Z = KDF(yX)$ and GK from X, Y , derives all $\{SK_j\}_{j \in \mathcal{S}}$ locally, and verifies the returned MAC_j values (either relayed by G or observed directly). An optional final acknowledgment from U completes the run.

4.3. Rationale and Differences

Perfect Forward Secrecy (PFS): Only the $U \leftrightarrow G$ path performs ephemeral ECDH; session material depends on Z and is therefore forward-secret even if s or any sk_j are later compromised.

Per-sensor isolation: A two-tier schedule is used: a short-lived group key GK expands to independent SK_j values via HKDF; each SK_j is delivered implicitly by opening $Wrap_j$. Compromise of one sensor does not expose other sensors' keys.

No XOR masking, no network-wide secret: The design removes any single value shared by all sensors. Identity protection and delivery authenticity are enforced via π_j and AEAD under sk_j .

Lightweight sensors: Sensors perform only hash/HMAC and AEAD (no elliptic-curve), fitting MCU-class devices.

Practical hardening: The augmented PAKE eliminates offline guessing under card theft; a stateless MACed cookie gates expensive work at G ; timestamps and per-session counters prevent replays; revocation/rekey are handled separately via LKH with subset wrapping as detailed and presented in Algorithm 3.

Algorithm 3: Revocation and Rekey (LOTUS-AKA.Rekey).

Input: Updated membership: active set S ; revoked set $\mathcal{R}_{\text{rev}}$; mode

Mode $\in \{\text{PerSensor}, \text{LKH}\}$

Output: Fresh session material for all $j \in S$; revoked nodes cannot derive new keys

State: At G : registry $\mathcal{R}$ with $\langle \text{SID}_j, sk_j \rangle$; (optional) LKH key tree $\mathcal{T}$ with node keys $\{K_v\}$; epoch e ; transcript hash t .

At S_j : long-term key sk_j (PerSensor mode); optionally a small set of LKH node keys on its path (LKH mode).

1 **(A) Membership update at G ;**

2 Remove all SID_r with $r \in \mathcal{R}_{\text{rev}}$ from the active set; define $S \leftarrow S \setminus \mathcal{R}_{\text{rev}}$;

3 **if** Mode = LKH **then**

4 For each revoked leaf r , update all ancestor node keys on the path to the root per LKH;
 let the updated tree be $\mathcal{T}'$;

5 Increment epoch $e \leftarrow e + 1$; pick fresh (TS, ctr);

6 **(B) Derive new group/session material;**

7 **if** piggybacked on a new run (Sec. 3.2) **then**

8 Compute ephemeral $Z \leftarrow KDF(yX)$; update t ;

9 **else**

// stand-alone rekey

10 Sample $R_e \leftarrow \{0, 1\}^{256}$; set $Z \leftarrow \text{HKDF}(\text{IKM} = R_e, \text{salt} = t, \text{info} = \text{"seed"} \parallel e)$;

11 Update $t \leftarrow H(t \parallel \text{"rekey"} \parallel e \parallel \text{TS})$;

12 Derive $\text{GK}' \leftarrow \text{HKDF}(\text{IKM} = Z, \text{salt} = t, \text{info} = \text{"GK"} \parallel e)$;

13 **(C) Distribute GK' to active sensors;**

Algorithm 3: Cont.

```

14 if Mode = PerSensor then
15     foreach  $j \in S$  do
16          $p_j \leftarrow \text{HMAC}_{sk_j}(\text{TS} \parallel \text{ctr})$ ; // per-run pseudonym
17          $\text{AD}_j \leftarrow \text{SID}_j \parallel \text{TS} \parallel t$ ;
18          $\text{Wrap}'_j \leftarrow \text{AEAD}_{sk_j}(\text{plaintext} = \text{GK}'; \text{AD} = \text{AD}_j)$ ;
19     Broadcast  $\{\text{TS}, \text{ctr}, (p_j, \text{Wrap}'_j)_{j \in S}\}$  (or include  $Y$  if tied to Sec. 3.2);
20 else // LKH subset-cover broadcast
21     Compute cover set  $\mathcal{C} \leftarrow \text{Cover}(\mathcal{T}', S)$  with minimal nodes  $\{v\}$  that cover exactly  $S$ ;
22     foreach  $v \in \mathcal{C}$  do
23          $\pi_v \leftarrow \text{HMAC}_{K_v}(\text{TS} \parallel \text{ctr})$ ; // pseudonym for node cover
24          $\text{AD}_v \leftarrow \text{NodeID}_v \parallel \text{TS} \parallel t$ ;
25          $\text{Wrap}'_v \leftarrow \text{AEAD}_{K_v}(\text{plaintext} = \text{GK}'; \text{AD} = \text{AD}_v)$ ;
26     Broadcast  $\{\text{TS}, \text{ctr}, (\pi_v, \text{Wrap}'_v)_{v \in \mathcal{C}}\}$ ;
27 (D) Sensor-side update;
28     PerSensor mode: Each  $S_j$  finds its tuple via  $p_j \stackrel{?}{=} \text{HMAC}_{sk_j}(\text{TS} \parallel \text{ctr})$ , opens  $\text{Wrap}'_j$ 
        with  $\text{AD}_j$ , and obtains  $\text{GK}'$ ;
29     LKH mode: Each  $S_j$  tests the few node keys it holds; for some cover  $v$ , verify  $\pi_v$  and
        open  $\text{Wrap}'_v$  with  $\text{AD}_v$  to obtain  $\text{GK}'$ ;
30     In both modes, derive per-sensor
         $\text{SK}'_j \leftarrow \text{HKDF}(\text{IKM} = \text{GK}', \text{salt} = t, \text{info} = \text{SID}_j \parallel \text{UID} \parallel t)$  and compute
         $\text{MAC}'_j \leftarrow \text{HMAC}_{\text{SK}'_j}(t)$ ;
31     Send  $\text{MAC}'_j$  to  $G$ ; gateways verify;
32 (E) Optional: forward-secure device keys;
33     If key evolution is enabled, after successful rekey set  $sk_j^{(e+1)} \leftarrow H(sk_j^{(e)})$  at  $S_j$  and in  $\mathcal{R}$ ;
        securely erase  $sk_j^{(e)}$ ;
34 Security notes;
35     Revoked sensors lack either  $sk_j$  (PerSensor) or updated node keys (LKH), so they cannot
        recover  $\text{GK}'$  nor compute any valid  $\text{MAC}'_j$ ;
36     Binding  $\text{SID}_j$  or  $\text{NodeID}_v$ ,  $(\text{TS}, \text{ctr})$ , and  $t$  into AD prevents cut-and-paste across
        epochs/runs;

```

Figure 2 depicts the one-to-many AKA run between the user U , the gateway G , and the sensor set $\{S_j\}$ (Please refer to Table 3 for the definition of the variables used in the one-to-many AKA run). First, U sends a single request to G carrying the PAKE finish tag, the response to a stateless DoS cookie, an (optionally anonymizing) subset tag for the addressed sensors, and the user's ephemeral $X = bP$. After validating the PAKE and cookie, G samples y , sets $Y = yP$, computes the ECDH secret $Z = \text{KDF}(yX)$, and derives a short-lived group key $\text{GK} = \text{HKDF}(Z, \tau)$, where τ is the running transcript hash. For each target sensor S_j , G then forms a pseudonym $\pi_j = \text{HMAC}_{sk_j}(\text{TS} \parallel \text{ctr})$ and a compact wrap $\text{Wrap}_j = \text{AEAD}_{sk_j}(\text{GK}; \text{AD} = \text{SID}_j \parallel \text{TS} \parallel \tau)$, and broadcasts a single message $\{Y, \text{TS}, \text{ctr}, (\pi_j, \text{Wrap}_j)_j\}$ to the whole set. Each S_j identifies its entry via π_j , opens Wrap_j using its long-term key sk_j to recover GK , derives an independent per-sensor key $\text{SK}_j = \text{HKDF}(\text{GK}, \text{SID}_j \parallel \text{UID} \parallel \tau)$, and returns a short confirmation $\text{MAC}_j = \text{HMAC}_{\text{SK}_j}(\tau)$. The user (directly or via G) verifies these MAC_j values after re-computing Z and GK from (X, Y) . Only U and G perform ephemeral ECDH (yielding PFS); sensors remain hash/AEAD-only. Because no network-wide secret is used and keys are derived per sensor, compromise of one device does not endanger others, and a single broadcast plus n short replies scales efficiently to large batches.

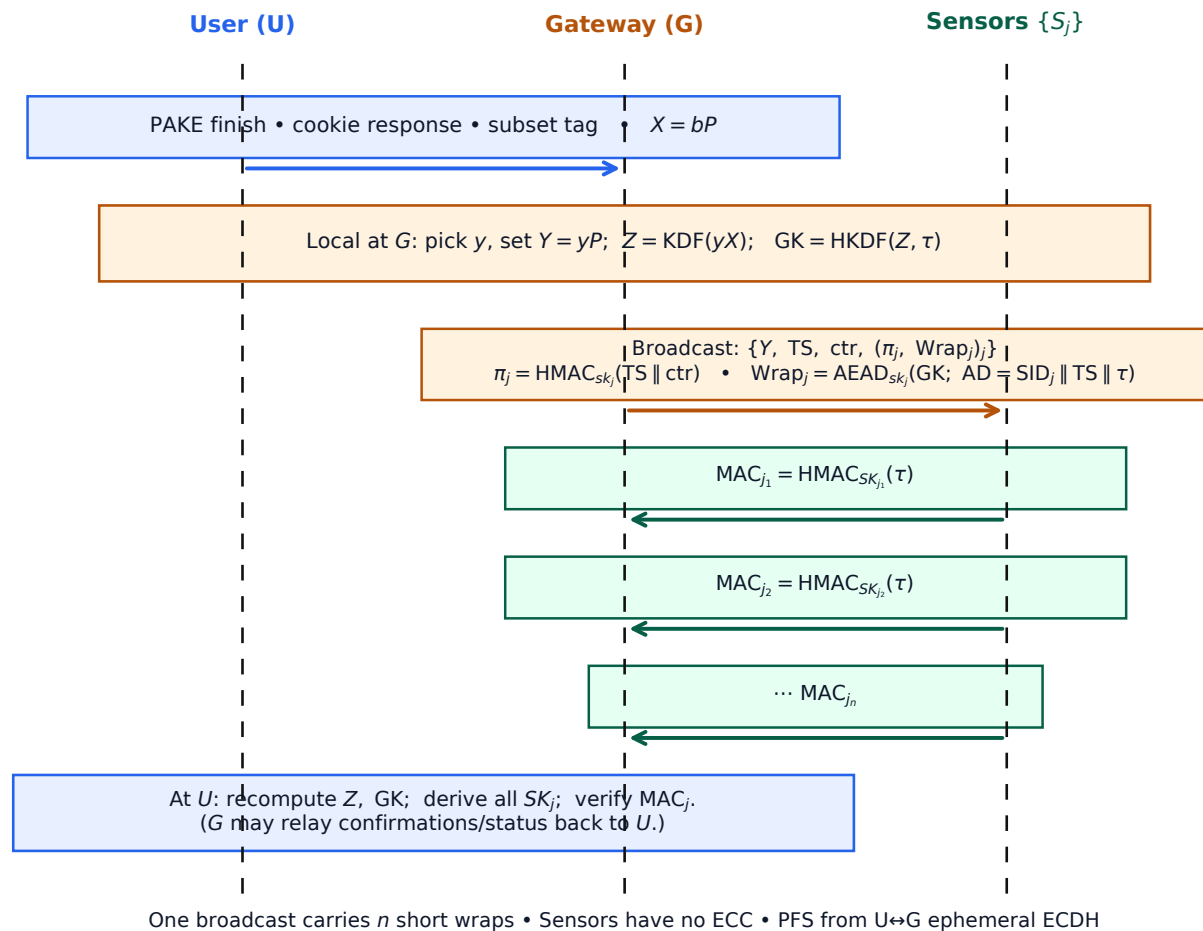


Figure 2. Protocol message flow. The user sends a single request to the gateway; the gateway performs ephemeral ECDH with the user to derive Z and GK , then broadcasts $\{Y, TS, \text{ctr}, (\pi_j, \text{Wrap}_j)_j\}$ to the sensor set. Each sensor S_j opens Wrap_j with SK_j , derives SK_j , and returns MAC_j .

Table 3. Key schedule and AEAD associated data (AD).

Label	Definition
Z	$\text{KDF}(bY) = \text{KDF}(yX)$ (ECDH secret)
GK	$\text{HKDF}(Z, \text{info} = \tau)$
SK_j	$\text{HKDF}(GK, \text{info} = \text{SID}_j \parallel \text{UID} \parallel \tau)$
π_j	$\text{HMAC}_{SK_j}(TS \parallel \text{ctr})$
Wrap_j	$\text{AEAD}_{SK_j}(GK; \text{AD} = \text{SID}_j \parallel TS \parallel \tau)$
AD fields	Identities, subset tag, TS, ctr, transcript hash τ

5. Security Analysis

This section sketches why LOTUS-AKA protocol meets its goals under standard assumptions. We use a game-based approach for AKE security and privacy, and complement it with symbolic verification.

5.1. Proof Roadmap (Game-Based)

We follow the standard computational model: a probabilistic polynomial-time (PPT) adversary is a (possibly non-uniform) randomized oracle machine $\mathcal{A}$ that, with the input of the security parameter 1^λ , runs in time $\text{poly}(\lambda)$ and interacts with the CK^+/eCK experiment via the usual oracles (Send, Reveal, EphemeralReveal, StateReveal, Corrupt, and Test).

All success probabilities/advantages are taken over the coins of the experiment and of $\mathcal{A}$; security requires every PPT adversary's advantage to be negligible in λ .

Before stating the theorems, it is beneficial to describe some parameters:

$\mathcal{B}$ denotes a probabilistic polynomial-time oracle machine that runs the AKE adversary $\mathcal{A}$ as a subroutine and leverages its view/queries to attack an underlying primitive. The quantity $\text{Adv}_{\mathcal{B}}^*$ is $\mathcal{B}$'s success/advantage in the corresponding security game (standard in reduction-based proofs).

λ : Security parameter; all algorithms are PPT in 1^λ .

$\text{Adv}_{\mathcal{A}}^{\text{ake}}$: Adversary $\mathcal{A}$'s AKE advantage in the CK^+ /eCK indistinguishability experiment (distinguishing a fresh session key from uniform under the standard oracle interface).

$\text{Adv}_{\mathcal{B}_1}^{\text{pake}}$: Advantage of some PPT reduction $\mathcal{B}_1$ in breaking the underlying PAKE security experiment (following BPR-style PAKE definitions).

$\text{Adv}_{\mathcal{B}_2}^{\text{prf}}(\text{HKDF})$: PRF advantage of $\mathcal{B}_2$ against the HMAC-based HKDF-Expand function (modeled as a keyed PRF on its input). Formally, $|\Pr[\mathcal{B}_2^F = 1] - \Pr[\mathcal{B}_2^R = 1]|$, where F is HKDF-Expand and R is a random function.

$\text{Adv}_{\mathcal{B}_3}^{\text{aead}}$: AEAD advantage of $\mathcal{B}_3$ against the symmetric cipher (aggregate of privacy IND-CPA/nonce-respecting and integrity INT-CTXT advantages for associated data).

$\text{Adv}_{\mathcal{B}_4}^{\text{cdh}}$: Success probability of $\mathcal{B}_4$ in solving the Computational Diffie–Hellman problem in the chosen group (given (g, g^a, g^b) , output g^{ab}).

$\text{negl}(\lambda)$: A negligible function in λ .

Theorem 1 (AKE security under CK^+ /eCK). *Let $\mathcal{A}$ be a PPT adversary in the CK^+ /eCK setting with access to Send, Reveal, EphemeralReveal, StateReveal, Corrupt, and Test oracles. Then, for any fresh session between U and S_j ,*

$$\text{Adv}_{\mathcal{A}}^{\text{ake}} \leq \text{Adv}_{\mathcal{B}_1}^{\text{pake}} + \text{Adv}_{\mathcal{B}_2}^{\text{prf}}(\text{HKDF}) + \text{Adv}_{\mathcal{B}_3}^{\text{aead}} + \text{Adv}_{\mathcal{B}_4}^{\text{cdh}} + \text{negl}(\lambda).$$

Proof. (i) Replace the augmented-PAKE transcript with an ideal functionality; any distinguishing advantage reduces to the PAKE security of the underlying scheme. (ii) Replace AEAD by an ideal (IND-CCA, INT-CTXT) channel; any forgery or decryption advantage reduces to AEAD security. (iii) Replace $Z = \text{KDF}(yX)$ by a uniform key using the CDH/X25519 hardness (via a standard DDH/CDH-style hybrid for ephemeral ECDH). (iv) Replace HKDF (extract&expand with SHA-256) by a PRF/RO; any bias in GK or SK_j becomes a PRF distinguisher. After these hops, the Test oracle returns a key computationally indistinguishable from random, yielding the bound. Freshness follows CK^+ /eCK conventions: revealing long-term user or sensor keys (or one side's ephemeral) without jointly exposing the session DH secret keeps the tested session fresh.

Let Hyb_0 be the real CK^+ /eCK AKE experiment and let Hyb_4 be the ideal one where the Test key is uniform. We transform Hyb_0 into Hyb_4 in four hops, each paid for by a standard advantage term; all oracles (Send/Reveal/EphemeralReveal/StateReveal/Corrupt/Test) and freshness are simulated exactly as in [39,40].

($\text{Hyb}_0 \rightarrow \text{Hyb}_1$ — PAKE idealization). Replace the real PAKE subroutine with its ideal experiment (BPR-style). A reduction $\mathcal{B}_1$ uses any AKE distinguisher to win the PAKE game, so

$$|\Pr[\text{Hyb}_0 \Rightarrow 1] - \Pr[\text{Hyb}_1 \Rightarrow 1]| \leq \text{Adv}_{\mathcal{B}_1}^{\text{pake}}.$$

(Hyb₁ → Hyb₂ — *HKDF as PRF*). Replace each call to HKDF-Expand by an oracle that is either the real PRF or a random function. A reduction $\mathcal{B}_2$ turns any change in $\mathcal{A}$'s success into a PRF win:

$$|\Pr[\text{Hyb}_1 \Rightarrow 1] - \Pr[\text{Hyb}_2 \Rightarrow 1]| \leq \text{Adv}_{\mathcal{B}_2}^{\text{prf}}(\text{HKDF}).$$

(Hyb₂ → Hyb₃ — *AEAD privacy and integrity*). Answer all protocol encryptions/decryptions using the AEAD challenger; any successful privacy distinction or ciphertext forgery by $\mathcal{A}$ provides an AEAD win to $\mathcal{B}_3$:

$$|\Pr[\text{Hyb}_2 \Rightarrow 1] - \Pr[\text{Hyb}_3 \Rightarrow 1]| \leq \text{Adv}_{\mathcal{B}_3}^{\text{aead}}.$$

(Hyb₃ → Hyb₄ — *Diffie–Hellman hardness*). Embed a CDH instance into the ephemeral DH exchange (e.g., set one party's ephemeral public key to a challenge point). If $\mathcal{A}$ still distinguishes the Test key from random in a fresh session, $\mathcal{B}_4$ extracts abP , solving CDH:

$$|\Pr[\text{Hyb}_3 \Rightarrow 1] - \Pr[\text{Hyb}_4 \Rightarrow 1]| \leq \text{Adv}_{\mathcal{B}_4}^{\text{cdh}}.$$

Finally, by the triangle inequality and since in Hyb₄ the Test key is uniform,

$$\text{Adv}_{\mathcal{A}}^{\text{ake}} = |\Pr[\text{Hyb}_0 \Rightarrow 1] - \frac{1}{2}| \leq \sum_{i=0}^3 |\Pr[\text{Hyb}_i \Rightarrow 1] - \Pr[\text{Hyb}_{i+1} \Rightarrow 1]| + \text{negl}(\lambda),$$

which yields

$$\text{Adv}_{\mathcal{A}}^{\text{ake}} \leq \text{Adv}_{\mathcal{B}_1}^{\text{pake}} + \text{Adv}_{\mathcal{B}_2}^{\text{prf}}(\text{HKDF}) + \text{Adv}_{\mathcal{B}_3}^{\text{aead}} + \text{Adv}_{\mathcal{B}_4}^{\text{cdh}} + \text{negl}(\lambda).$$

The negligible term covers standard bad events (e.g., collisions, and freshness failure) under our assumptions. $\square$

Theorem 2 (Post-compromise secrecy of Z , GK , and SK_j). *Let U and S_j complete a fresh session producing the shared Diffie–Hellman value $Z = xyP \in \mathbb{G}$ and derive keys via $\text{GK} \leftarrow \text{HKDF}(Z; \text{info} = \tau)$ and $\text{SK}_j \leftarrow \text{HKDF}(\text{GK}; \text{info} = \text{sk} \parallel \text{SID}_j \parallel \tau)$. If, after this completion, the gateway master secret s is revealed, then for any PPT adversary $\mathcal{A}$ in the CK^+/eCK model, the AKE distinguishing advantage satisfies*

$$\text{Adv}_{\mathcal{A}}^{\text{ake}} \leq \text{Adv}_{\mathcal{B}_4}^{\text{cdh}} + \text{Adv}_{\mathcal{B}_2}^{\text{prf}}(\text{HKDF}) + \text{negl}(\lambda),$$

hence, Z , GK , and all SK_j from that session remain indistinguishable from random.

Proof. We use a standard sequence of games. Let λ be the security parameter, and let $\text{negl}(\lambda)$ denote a negligible function.

Game G_0 (Real): This is the real CK^+/eCK experiment for $\mathcal{A}$ with a fresh test session. Let $\text{Adv}_0 = |\Pr[b = 1] - \frac{1}{2}|$ be $\mathcal{A}$'s distinguishing advantage.

Game G_1 (Enforce freshness): Identical to G_0 , except the game aborts if $\mathcal{A}$ violates freshness (e.g., reveals the test session's long-term or ephemeral secrets). Because the tested session is fresh by hypothesis, $\Pr[G_0 \neq G_1] = 0$ and $\text{Adv}_1 = \text{Adv}_0$.

Game G_2 (Hide Z by hardness of CDH): We change only how keys are conceptually computed as follows: keep the public transcript $(P, X = xP, Y = yP, \tau, \text{SID}_j)$ exactly as in G_1 , but regard the seed fed to HKDF as unknown to $\mathcal{A}$ unless it can compute $Z = xyP$ from (P, X, Y) . By the computational Diffie–Hellman assumption, any PPT algorithm that outputs Z from (P, X, Y) has advantage at most $\text{Adv}_{\mathcal{B}_4}^{\text{cdh}}$. Equivalently, conditioned on the

transcript, Z is computationally unpredictable; thus, Z retains (computational) min-entropy $\kappa = \Theta(\lambda)$. Therefore the transition $G_1 \Rightarrow G_2$ changes $\mathcal{A}$'s view by at most $\text{Adv}_{\mathcal{B}_4}^{\text{cdh}}$:

$$|\text{Adv}_1 - \text{Adv}_2| \leq \text{Adv}_{\mathcal{B}_4}^{\text{cdh}}.$$

Game G_3 (Replace HKDF outputs by random via PRF security): Since in G_2 the HKDF input has high (computational) entropy unknown to $\mathcal{A}$, we use that HKDF (Extract-then-Expand with HMAC) is a pseudorandom function/extractor. Replace HKDF($\cdot$; info) at all points (for GK and for each SK_j) by a truly random function with the same interface. By the PRF security of HKDF, this changes $\mathcal{A}$'s advantage by at most $\text{Adv}_{\mathcal{B}_2}^{\text{prf}}(\text{HKDF})$; any residual statistical slack from extraction (e.g., collision or leftover-hash-style terms) is absorbed in $\text{negl}(\lambda)$:

$$|\text{Adv}_2 - \text{Adv}_3| \leq \text{Adv}_{\mathcal{B}_2}^{\text{prf}}(\text{HKDF}) + \text{negl}(\lambda).$$

Concluding bound: In G_3 , the test key is independent of the transcript and uniformly random, so $\text{Adv}_3 = 0$. By the triangle inequality over the hops $G_0 \rightarrow G_1 \rightarrow G_2 \rightarrow G_3$,

$$\begin{aligned} \text{Adv}_{\mathcal{A}}^{\text{ake}} = \text{Adv}_0 &\leq \underbrace{|\text{Adv}_0 - \text{Adv}_1|}_{=0} + \underbrace{|\text{Adv}_1 - \text{Adv}_2|}_{\leq \text{Adv}_{\mathcal{B}_4}^{\text{cdh}}} + \underbrace{|\text{Adv}_2 - \text{Adv}_3|}_{\leq \text{Adv}_{\mathcal{B}_2}^{\text{prf}}(\text{HKDF}) + \text{negl}(\lambda)} \\ &\leq \text{Adv}_{\mathcal{B}_4}^{\text{cdh}} + \text{Adv}_{\mathcal{B}_2}^{\text{prf}}(\text{HKDF}) + \text{negl}(\lambda). \end{aligned}$$

Finally, revealing the gateway master secret s *after* completion does not help compute Z (since x, y are erased), nor distinguish HKDF outputs in G_3 ; hence, Z , GK, and every SK_j from that session remain indistinguishable from random. $\square$

Theorem 3 (Privacy). *User anonymity and sensor unlinkability hold against a network adversary.*

Proof. User identity is hidden by the augmented-PAKE (server-aided, no verifier for offline guessing). Sensors appear only via per-session pseudonyms $\pi_j = \text{HMAC}_{sk_j}(\text{TS} \parallel \text{ctr})$; without sk_j , π_j is a PRF output that changes with (TS, ctr) , preventing cross-run linkage. SID_j is never sent in the clear text when the subset tag is anonymized; associated data (AD) binds identities to ciphertexts without revealing them. $\square$

Theorem 4 (Resilience). *Against a Dolev–Yao network adversary with access to Send, Reveal, EphemeralReveal, StateReveal, and Corrupt oracles (CK^+/eCK freshness), the protocol resists (i) key-compromise impersonation (KCI), (ii) replay, (iii) man-in-the-middle and transcript-splicing, (iv) party impersonation without credentials, and (v) offline dictionary attacks under smart-card theft. It also enforces ciphertext integrity for per-sensor wraps and confirmations.*

Proof. We argue each property under the assumptions used in Theorem 1 (augmented-PAKE security, AEAD IND-CCA and INT-CTXT, CDH hardness for X25519/P-256, and HKDF as a PRF).

(i) *KCI resistance.* Suppose the adversary learns the long-term credential of U (or a sensor S_j) and attempts to impersonate an honest peer to that corrupted party. A successful attack would require producing a session key consistent with an accepting transcript without knowledge of the fresh ECDH secret $Z = \text{KDF}(yX)$. In CK^+/eCK , a session is fresh if the adversary does not obtain both parties' matching ephemeral secrets. Replacing Z by uniform (CDH hardness) and HKDF by a PRF (Theorem 1 hybrids) makes the test key indistinguishable from random unless the adversary breaks the augmented-PAKE finish to

force acceptance on bogus ephemerals, which contradicts PAKE security. Thus, learning a party's long-term key does not enable impersonation of another honest party to it.

(ii) *Replay resistance.* Each run binds a timestamp and counter (TS, ctr) into the transcript hash τ and into AEAD associated data. Replaying an old broadcast or an old confirmation MAC_j would require either acceptance of a stale (TS, ctr) outside the window (rejected by the receiver) or the ability to alter (TS, ctr, τ) without detection, which would violate AEAD integrity or the PRF security of HMAC/HKDF.

(iii) *Man-in-the-middle and transcript splicing.* An active adversary cannot bias or splice the key schedule without detection: Z is derived from fresh (b, y) ; replacing X or Y to force a chosen Z reduces to solving CDH. Moreover, associated data authenticates SID_j , (TS, ctr), and τ , binding identities and the full transcript to each ciphertext; any splice or mix-and-match breaks AEAD INT-CTXT.

(iv) *Party impersonation.* To impersonate the user to G , the adversary must complete the augmented-PAKE; by assumption, this is infeasible without the password (or an online attack that is rate-limited). To impersonate a sensor to U or G , the adversary must return a valid $\text{MAC}_j = \text{HMAC}_{SK_j}(\tau)$ where SK_j is derived from GK recovered by decrypting Wrap_j . Forging such a confirmation without sk_j or SK_j violates AEAD IND-CCA (to learn GK) or the PRF security of HKDF/HMAC (to compute SK_j or the MAC).

(v) *Smart-card theft and offline dictionary.* The login uses an augmented PAKE (OPAQUE-style): the server stores a hardened record and the card stores only an opaque envelope. A copied card or server record alone yields no offline password oracle; any guessing requires online interaction and is detected/rate-limited, by the PAKE security definition.

Ciphertext integrity. Per-sensor wraps Wrap_j and confirmations MAC_j are protected, respectively, by AEAD INT-CTXT and by HMAC as a PRF; undetected modification or forgery would contradict these assumptions. $\square$

5.2. Symbolic Verification

We model the protocol in ProVerif (and Tamarin for sanity checks) with standard constructors for tuples, HMAC, HKDF, and AEAD (idealized as Authenticated Encryption with explicit AD). The model includes an active Dolev–Yao adversary, device-capture events, and cookie issuance/validation. We verify

- Secrecy of SK_j for honest U , G , and any S_j not explicitly corrupted as follows: query $\text{attacker}(SK_j) == \text{false}$.
- Authentication correspondence between U and each addressed S_j : event $\text{end_U}(j)$ implies event $\text{end_Sj}(j)$ with matching transcript hash τ .
- Unlinkability of sensor sessions: observational equivalence between two traces that swap the identities of two honest sensors while preserving the multiset of messages.

The models reflect the optional key-evolving variant for forward-secure wraps: sk_j is advanced per epoch and previous values are erased, enabling queries about past sessions after a capture event at epoch e^* .

5.3. DoS and Availability

The gateway issues a stateless cookie (MAC over source IP/tuple, coarse timestamp, and a brief salt) that must be echoed by U before any AEAD processing or per-sensor work, implementing MAC-before-work. Parsing and MAC checks are constant-time with length and range validation to avoid parser bombs. Replays are bounded by a sliding window over (TS, ctr), with transcript binding in AD blocking cut-and-paste across runs. On sensor nodes, a single cheap HMAC check over the cookie/pseudonym precedes state allocation, ensuring that unauthenticated traffic cannot force expensive operations or buffer exhaustion.

The base design attains PFS against server-side compromise and best-possible security for symmetric-only sensors; enabling the (one-line) key-evolving wrap hardens against post-compromise of sensors without adding public-key costs to devices. Overall, the bounds in Theorems 1–4 reduce attacks to well-studied assumptions (augmented-PAKE, AEAD, CDH/X25519, and HKDF-as-PRF) while keeping device computation strictly symmetric.

6. Performance Evaluation

We evaluate the protocol along operation counts (Table 4), bytes on the wire, and end-to-end latency/energy on MCU-class sensors and a commodity gateway. Our focus is the one-to-many setting where a user addresses n sensors in a single run.

Table 4. LOTUS-AKA vs. one-to-many baselines (per run with n addressed sensors).

Scheme	Sensor Work (per S_j)	Gateway Work (per Run)	Bytes-on-Air (Control Plane)	Notes
B3: Broadcast auth only	MAC verify (+buffering)	MAC generation (+key schedule)	$O(1)$ broadcast (plus disclosure metadata); no ACKs	Authenticity only; no Key Agreement; no per-sensor session keys
B1: Shared GK broadcast	$1 \times \text{AEAD_open}$ (+optional HKDF)	$1 \times \text{AEAD_enc}$	Broadcast $\approx L_{\text{hdr}} + (L_{GK} + L_{\text{tag}} + L_{\text{nonce}})$; ACKs optional	All receivers share the same GK ; limited sensor-to-sensor isolation
B2: LKH subset rekey (shared GK)	$\leq 1 \times \text{AEAD_open}$; state $O(\log N)$	$ cover(S) \times \text{AEAD_enc}$	Broadcast $\approx L_{\text{hdr}} + cover(S) \cdot (L_{GK} + L_{\text{tag}} + L_{\text{nonce}})$; ACKs optional	Efficient subset distribution; delivered key still shared across the addressed subset
LOTUS-AKA	HMAC (pseudonym) + AEAD_open + HKDF + HMAC (confirm)	$2 \times \text{ECC_mult} + n \times (\text{HMAC} + \text{AEAD_enc} + \text{HMAC})$	Broadcast $\approx L_{\text{hdr}} + n \cdot L_{\text{tuple}}$; ACKs $\approx n \cdot (L_{\text{mac}} + \text{hdr})$	Perfect Forward Secrecy on U-G; per-sensor key isolation; sensors run symmetric-only

6.1. Metrics and Setup

Metrics: We count (i) elliptic-curve scalar multiplications (#ECC_mult) performed only by U and G ; (ii) hashes/HMACs (#HMAC/#Hash); (iii) authenticated encryptions/decryptions (#AEAD); and (iv) total bytes transmitted. For latency/energy, we report per-phase time (auth handshake, per-sensor wrap open) and the aggregate vs. n .

Platforms: Sensors are ARM Cortex-M0 and Cortex-M4 class MCUs (typical: 48–80 MHz, 16–256 KB RAM, 64–512 KB Flash) using constant-time HMAC-SHA256 and either AES-GCM or ChaCha20-Poly1305. The gateway is an x86_64 class CPU with hardware AES support. All code paths use fixed-length parsing and *branch-free* verification.

6.2. Microbenchmarks

We isolate (i) the $U \leftrightarrow G$ ephemeral ECDH/OPAQUE finish (auth phase); (ii) per-sensor wrap open on S_j (HMAC pseudonym test, AEAD open, HKDF-expand, and confirmation MAC). Let $t_{\text{AEAD-open}}$, t_{HMAC} , and t_{HKDF} denote per-operation times on the MCU; for a batch of n sensors, the sensor-side cost is

$$T_{\text{sensor}}(n) \approx t_{\text{HMAC}} + t_{\text{AEAD-open}} + t_{\text{HKDF}} + t_{\text{HMAC}} \quad (\text{per } S_j),$$

while the user-side batch work after the ECDH is $\approx n \cdot (t_{\text{HKDF}} + t_{\text{HMAC}})$ to derive $\{SK_j\}$ and verify $\{\text{MAC}_j\}$. Energy follows $E = \sum_i I_{\text{active}} \cdot V \cdot t_i$ for each primitive i ; we report both latency and energy as functions of n .

6.3. One-to-Many Baselines

In addition to the repeated one-to-one baseline (invoking a full 1→1 handshake n times), we compare LOTUS-AKA against representative lightweight one-to-many families that are commonly used as design patterns in IoT/WSN deployments. We instantiate these baselines using the same primitive choices and message-format conventions as our implementation (AEAD, HMAC/HKDF, and identical byte encodings), and we report the following metrics: sensor computation, gateway computation, bytes on air, and end-to-end latency (see Table 4).

6.3.1. B1: Shared Group-Key Broadcast (Single GK for All)

The gateway distributes one group session key GK that is shared by all receivers in the addressed set (or even all sensors), using a single AEAD-encrypted payload under a group credential. Sensors open the ciphertext to obtain GK and then (optionally) derive local traffic keys from GK (please see Table 5).

Table 5. Operation counts per party (one run with n sensors).

Party	Operations
User U	#ECC_mult = 2 (public key, DH shared secret); HKDF: 1 extract + 1 expand; per- j : HKDF-expand + HMAC (verify MAC $_j$).
Gateway G	#ECC_mult = 2; PAKE verify; cookie MAC; per- j : HMAC (pseudonym π_j) + AEAD_enc(Wrap $_j$) + HMAC (confirm).
Sensor S_j	HMAC (cookie/pseudonym) + AEAD_open + HKDF-expand + HMAC (confirm).

6.3.2. B2: LKH Subset Rekey (Distribute a Shared GK)

The gateway maintains a Logical Key Hierarchy (LKH) key tree and distributes a fresh shared GK to an addressed subset using an encrypted rekey messages. Each sensor stores $O(\log N)$ long-term node keys and opens at most one relevant rekey ciphertext to obtain GK .

6.3.3. B3: Broadcast Authentication Only (no AKA)

This family provides sender/broadcast authenticity with symmetric primitives and delayed disclosure or equivalent techniques, but does not establish per-sensor session keys via Key Agreement. We include it as an authenticity-only reference point for control-plane broadcasts (please see Table 6).

$$\text{Let } L_{\text{hdr}} = L_Y + L_{\text{ts}} + L_{\text{ctr}} \text{ and } L_{\text{tuple}} = L_{\pi} + (L_{GK} + L_{\text{tag}} + L_{\text{nonce}}).$$

Table 6. Communication cost vs. n (broadcast + n acks).

Term	Size (bytes)
Y (ephemeral pub. key)	L_Y (e.g., 32 for X25519).
$TS \parallel \text{ctr}$	$L_{\text{ts}} + L_{\text{ctr}}$ (e.g., 8 + 4).
Per-sensor π_j	L_{π} (e.g., 16 truncated HMAC).
Per-sensor Wrap $_j$	$L_{GK} + L_{\text{tag}} + L_{\text{nonce}}$ (e.g., 32 + 16 + 12).
Total Broadcast	$L_Y + L_{\text{ts}} + L_{\text{ctr}} + n \cdot (L_{\pi} + L_{GK} + L_{\text{tag}} + L_{\text{nonce}})$
Total Acks	$n \cdot (L_{\text{mac}} + \text{headers})$ (e.g., 16 + headers).

6.4. Comparison with Baselines

Prior one-to-many schemes that derive a single group key and reuse it across sensors and a repeated one-to-one baseline that invokes a full handshake n times were studied. Our design keeps #ECC_mult constant in n (only U and G do ECDH once) and shifts scaling to cheap symmetric operations. Communication is one broadcast of size $O(n + 1)$ and n short confirmations; the one-to-one baseline sends $O(n)$ full handshakes. The results (Figure 3)

show linear growth with a small slope for ours (dominated by one AEAD wrap per sensor), versus a much steeper slope when full ECDH+PAKE is repeated n times.

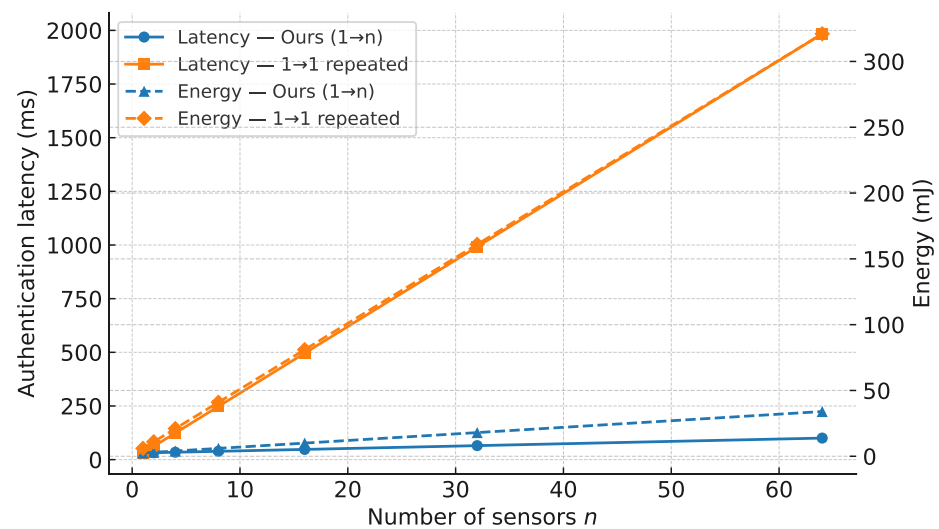


Figure 3. Authentication latency (left axis) and energy (right axis) vs. n . Our one-to-many run performs one ECDH/PAKE and n symmetric wraps; the 1-to-1 baseline repeats ECDH/PAKE n times.

6.5. Variants and Sensitivity Analysis

We evaluate how specific design choices affect performance and robustness: (i) PAKE off (replaced with a pre-shared credential) to quantify login overhead; (ii) AEAD choice (AES-GCM vs. ChaCha20-Poly1305) to capture MCU/CPU asymmetries; (iii) Cookie off to measure gateway load and drop rate under adversarial floods. These what-if experiments isolate each component's contribution to latency, energy, and availability.

Since sensors avoid ECC entirely, their per-sensor work is stable and predictable (one AEAD open, a few HMAC/HKDF evaluations). The gateway performs batched symmetric work with one ECDH per run. As n grows, communication dominates; our broadcast+ack pattern maintains low overhead per sensor and avoids n network handshakes. On Cortex-M0/M4, the slope of $T_{\text{sensor}}(n)$ is driven chiefly by AEAD_{open}; choosing ChaCha20-Poly1305 on M0 (no AES) and AES-GCM on M4/M7 (with hardware AES) yields the best latency/energy across the tested configurations.

6.6. Communication Cost vs. n (Figure 4)

Our curve grows linearly with a small slope because the broadcast has a small fixed header ($Y+TS+ctr$) and then adds one short tuple (π_j, Wrap_j) per sensor plus a 16 B confirmation. With the parameters used to generate the figure, this is roughly $44 + 92n$ B. The $1 \rightarrow 1$ baseline must ship a full PAKE/ECDH transcript per sensor (modeled ≈ 600 B) plus a small confirmation, hence the much steeper slope. At $n = 64$, the gap is ~ 6 KB vs. ~ 39 KB.

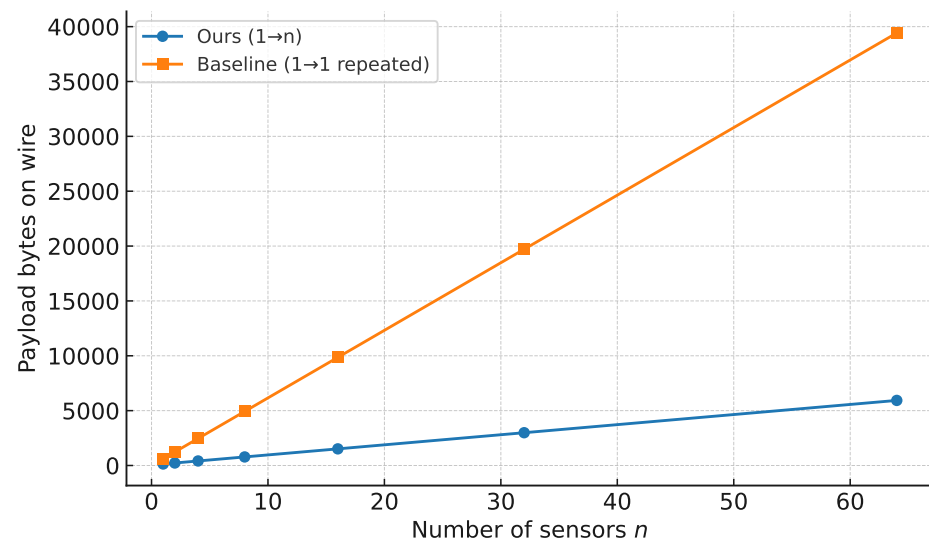


Figure 4. Payload bytes vs. number of sensors n : our one-to-many run (one broadcast + n short acks) vs. a repeated $1 \rightarrow 1$ baseline (full handshake n times).

6.7. AEAD Choice on MCUs (Figure 5)

On Cortex-M0 (no AES acceleration), ChaCha20-Poly1305 is faster (≈ 1.3 ms) than AES-GCM (≈ 2.8 ms). On Cortex-M4 with AES support, AES-GCM is best (≈ 0.62 ms) and ChaCha20-Poly1305 is slightly slower (≈ 1.12 ms). This supports a simple deployment rule: use ChaCha on M0-class boards; use AES-GCM on M4/M7 with hardware AES.

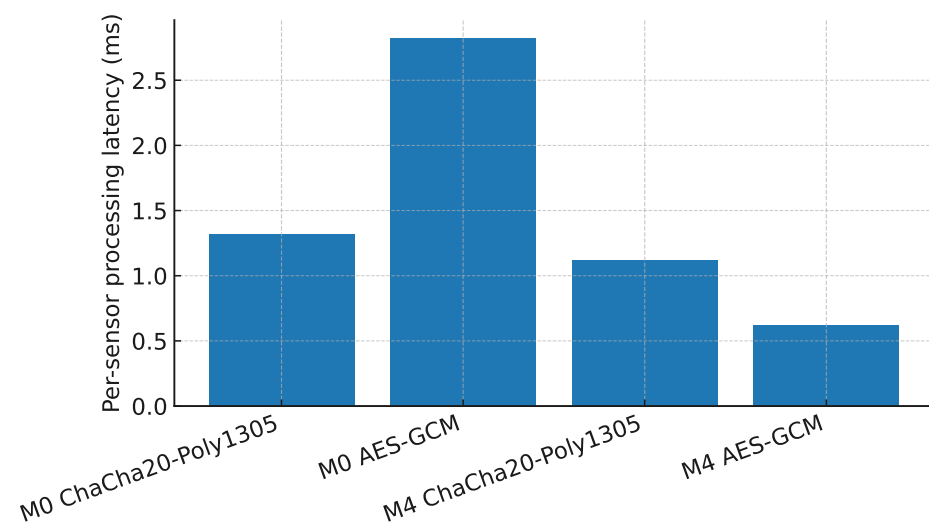


Figure 5. Per-sensor processing latency on Cortex-M0/M4 for AES-GCM and ChaCha20-Poly1305 (including HMAC pseudonym, AEAD open, HKDF-expand, and confirmation MAC).

6.8. DoS Resilience Under Adversarial Load (Figure 6)

The stateless cookie cheaply filters unauthenticated floods, so the gateway sustains close to its handshake cap even when attack traffic grows. Without the cookie, expensive parsing of junk requests starves the CPU and accepted sessions drop toward zero as the attack rate increases.

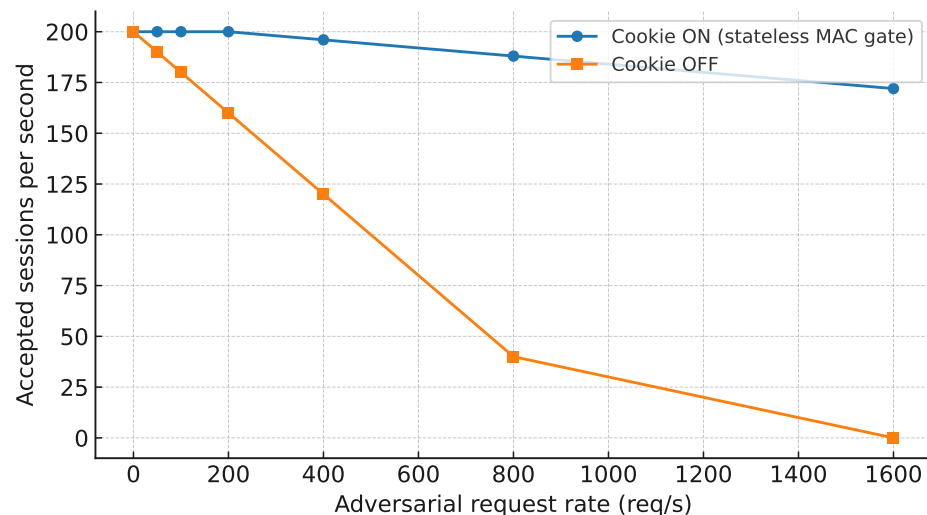


Figure 6. Accepted sessions per second vs. adversarial request rate. With a stateless cookie (MAC-before-work), throughput remains near the gateway’s capacity; disabling the cookie collapses throughput under flood.

6.9. Authentication Latency and Energy vs. n (Figure 3)

Because ECDH/PAKE is executed once per run (by U and G), our latency/energy increases gently with n , dominated by one AEAD open and a few hashes per sensor. The repeated $1 \rightarrow 1$ baseline redoes ECDH/PAKE for every sensor, so both latency and energy grow much faster with n .

7. Conclusions

This paper revisits the problem of authenticating a user to a *set* of constrained sensors and deriving fresh, confidential session keys in one shot. We identify concrete gaps in prior “lightweight” proposals—lack of true PFS, reliance on a network-wide secret, per-batch group keys that leak across sensors, offline guessing under smart-card theft, and weak DoS defenses—and address them with a design that keeps sensors strictly symmetric while moving ephemeral public-key work to the user and gateway.

The proposed LOTUS-AKA protocol performs a single ephemeral ECDH between U and G to derive Z and a transcript-bound group key GK via HKDF. The gateway then broadcasts one message that contains per-sensor, AEAD-wrapped material under long-term sk_j , and short pseudonyms π_j to let each sensor privately recognize itself. Each sensor opens exactly one AEAD, runs a few HMAC/HKDF calls, and returns a confirmation MAC; no ECC is required on devices. Login uses an augmented PAKE, eliminating offline dictionary attacks even if a card or server record is stolen. A stateless cookie enforces MAC-before-work at the gateway and sensors, mitigating floods without maintaining per-source state. Revocation and rekey are supported via subset wrapping (and optionally LKH), while an optional key-evolving variant for sk_j strengthens post-compromise secrecy on devices that can securely erase.

We analyze security in the CK^+ /eCK style: AKE security reduces to the PAKE, AEAD, HKDF-as-PRF, and CDH/X25519 assumptions; PFS holds for server compromise and can be extended to sensor compromise with key evolution; privacy follows from PAKE-hidden user identifiers and per-session pseudonyms. A symbolic ProVerif/Tamarin model captures secrecy, correspondence, and unlinkability queries, including device capture and the cookie gate.

Our evaluation indicates that one ECDH per run plus n symmetric wraps yields latency and energy that grow gently with the number of sensors, while a repeated $1 \rightarrow 1$

baseline grows much faster. Communication similarly scales linearly with a small per-sensor increment. Microbenchmarks suggest choosing ChaCha20-Poly1305 on Cortex-M0 (no AES) and AES-GCM on Cortex-M4/M7 (with AES) for best per-sensor latency. Under adversarial load, the cookie preserves near-capacity throughput, whereas disabling it causes accepted sessions to collapse.

Overall, the design provides a practical, analyzable path to one-to-many AKA with strong security properties and deployable costs on Cortex-M-class devices.

Future work includes completing a full mechanized proof of the game-based reductions; integrating precise energy measurements from hardware boards; studying loss/retransmission on low-power links; tightening revocation at scale (e.g., LKH variants with constrained memory); and exploring post-quantum KEM drop-ins for the $U \leftrightarrow G$ ephemeral while keeping sensors symmetric.

Author Contributions: Conceptualization, H.E.G. and A.A.-K.; Methodology, H.E.G., A.H.E.F., A.M. and A.N.; Software, H.E.G. and A.H.E.F.; Validation, H.E.G., A.M. and A.A.-K.; Formal analysis, A.H.E.F., A.A.-K. and A.N.; Investigation, H.E.G. and A.H.E.F.; Resources, A.H.E.F.; Data curation, A.A.-K.; Writing—original draft, H.E.G. and A.N.; Writing—review and editing, A.H.E.F., A.M., A.A.-K. and A.N.; Visualization, A.A.-K.; Supervision, A.M. and A.N.; Project administration, A.M.; Funding acquisition, A.M. and A.N. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The raw data supporting the conclusions of this article will be made available by the authors on request.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Perwej, Y.; Haq, K.; Parwej, F.; Mumdouh, M.; Hassan, M. The internet of things (IoT) and its application domains. *Int. J. Comput. Appl.* **2019**, *975*, 182. [\[CrossRef\]](#)
2. Makhdoom, I.; Abolhasan, M.; Lipman, J.; Liu, R.P.; Ni, W. Anatomy of threats to the internet of things. *IEEE Commun. Surv. Tutor.* **2018**, *21*, 1636–1675. [\[CrossRef\]](#)
3. Kang, B.; Kim, D.; Choo, H. Internet of everything: A large-scale autonomic IoT gateway. *IEEE Trans.-Multi-Scale Comput. Syst.* **2017**, *3*, 206–214. [\[CrossRef\]](#)
4. Hasan, M.K.; Weichen, Z.; Safie, N.; Ahmed, F.R.A.; Ghazal, T.M. A survey on key agreement and authentication protocol for internet of things application. *IEEE Access* **2024**, *12*, 61642–61666. [\[CrossRef\]](#)
5. Fiterau-Brostean, P.; Jonsson, B.; Merget, R.; De Ruiter, J.; Sagonas, K.; Somorovsky, J. Analysis of DTLS implementations using protocol state fuzzing. In Proceedings of the SEC'20: 29th USENIX Conference on Security Symposium, Berkeley, CA, USA, 12–14 August 2020; pp. 2523–2540.
6. Walz, A.; Sikora, A. Exploiting dissent: Towards fuzzing-based differential black-box testing of TLS implementations. *IEEE Trans. Dependable Secur. Comput.* **2017**, *17*, 278–291. [\[CrossRef\]](#)
7. Pan, Y.; Lin, W.; He, Y.; Zhu, Y. Coverage-guided differential testing of TLS implementations based on syntax mutation. *PLoS ONE* **2022**, *17*, e0262176. [\[CrossRef\]](#) [\[PubMed\]](#)
8. Gbur, K.Y.; Tschorsch, F. QUICforge: Client-side Request Forgery in QUIC. In Proceedings of the Network and Distributed System Security (NDSS) Symposium 2023, San Diego, CA, USA, 27 February–3 March 2023.
9. Maleh, Y.; Ezzati, A.; Belaissaoui, M. Dos attacks analysis and improvement in dtls protocol for internet of things. In Proceedings of the International Conference on Big Data and Advanced Wireless Technologies, Blagoevgrad, Bulgaria, 10–11 November 2016; pp. 1–7.
10. Chatzoglou, E.; Kouliaridis, V.; Karopoulos, G.; Kambourakis, G. Revisiting QUIC attacks: A comprehensive review on QUIC security and a hands-on study. *Int. J. Inf. Secur.* **2023**, *22*, 347–365. [\[CrossRef\]](#)

11. Wazid, M.; Das, A.K.; Odelu, V.; Kumar, N.; Conti, M.; Jo, M. Design of secure user authenticated key management protocol for generic IoT networks. *IEEE Internet Things J.* **2018**, *5*, 269–282. [\[CrossRef\]](#)
12. Vinoth, R.; Deborah, L.J.; Vijayakumar, P.; Das, M.L.; Jhanjhi, N.Z.; Alazab, M. Secure multifactor authenticated key agreement scheme for industrial IoT. *IEEE Internet Things J.* **2021**, *8*, 3801–3811. [\[CrossRef\]](#)
13. Algubili, B.H.T.; Kumar, N.; Lu, H.; Yassin, A.A.; Boussada, R.; Mohammed, A.J.; Liu, H. EPSAPI: An efficient and provably secure authentication protocol for an IoT application environment. *Peer-to-Peer Netw. Appl.* **2022**, *15*, 2179–2198. [\[CrossRef\]](#)
14. Qin, Y.; Xu, M.; Zhao, Y.; Li, B. A one-to-many authentication and key agreement scheme based on the sliding window. In Proceedings of the Fourth International Conference on High Performance Computing and Communication Engineering (HPCCE 2024), Suzhou, China, 22–24 November 2025; Volume 13630, p. 1363002.
15. Ming, Y.; Yang, P.; Mahdikhani, H.; Lu, R. A secure one-to-many authentication and key agreement scheme for industrial IoT. *IEEE Syst. J.* **2023**, *17*, 2225–2236. [\[CrossRef\]](#)
16. Aly, A.H.; Ghalwash, A.T.E.F.; Nasr, M.O.N.A.; El-Hafez, A.A. A new lightweight authenticated key agreement protocol for IoT in cloud computing. *J. Eng. Sci. Technol.* **2021**, *16*, 3987–4005.
17. Daniel, R.M.; Thomas, A.; Rajsingh, E.B.; Silas, S. A Strengthened eCK Secure Identity Based Authenticated Key Agreement Protocol based on the Standard CDH Assumption. *Inf. Comput.* **2023**, *294*, 105067. [\[CrossRef\]](#)
18. Meier, S.; Schmidt, B.; Cremers, C.; Basin, D. The TAMARIN Prover for the Symbolic Analysis of Security Protocols. In *Proceedings of CAV 2013 (LNCS 8044)*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 696–701. [\[CrossRef\]](#)
19. Rescorla, E.; Modadugu, N. RFC 6347: *Datagram Transport Layer Security Version 1.2*; RFC Editor: USA, 2012. Available online: <https://www.rfc-editor.org/rfc/rfc6347.html> (accessed on 22 December 2025).
20. Rescorla, E.; Tschofenig, H.; Modadugu, N. RFC 9147: *The Datagram Transport Layer Security (DTLS) Protocol Version 1.3*; RFC Editor: USA, 2022. Available online: <https://datatracker.ietf.org/doc/rfc9147/> (accessed on 22 December 2025).
21. Selander, G.; Mattsson, J.; Palombini, F.; Seitz, L. RFC 8613: *Object Security for Constrained RESTful Environments (OSCORE)*; RFC Editor: USA, 2019. Available online: <https://datatracker.ietf.org/doc/rfc8613/> (accessed on 22 December 2025).
22. Selander, G.; Preuß Mattsson, J.; Palombini, F. RFC 9528: *Ephemeral Diffie–Hellman Over COSE (EDHOC)*; RFC Editor: USA, 2024. Available online: <https://datatracker.ietf.org/doc/rfc9528/> (accessed on 22 December 2025).
23. Perez, E.L.; Seler, G.; Mattsson, J.P.; Watteyne, T.; Vučinić, M. EDHOC Is a New Security Handshake Standard: An Overview of Security Analysis. *Computer* **2024**, *57*, 101–110. [\[CrossRef\]](#)
24. Palombini, F.; Tiloca, M.; Höglund, R.; Hristozov, S.; Selander, G. RFC 9668: *Using Ephemeral Diffie–Hellman Over COSE (EDHOC) with the Constrained Application Protocol (CoAP) and Object Security for Constrained RESTful Environments (OSCORE)*; RFC Editor: USA, 2024. Available online: <https://www.rfc-editor.org/rfc/rfc9668.html> (accessed on 22 December 2025).
25. Weng, Y.; Zhang, Y. A survey of secure time synchronization. *Appl. Sci.* **2023**, *13*, 3923. [\[CrossRef\]](#)
26. Zheng, Z.; Li, X.; Wang, H.; Wu, G.; Zhang, Y. Research on SSL/TLS Security Differences Based on RFC Documents. In Proceedings of the 2024 International Conference on Computing, Networking and Communications (ICNC), IEEE Computer Society, Big Island, HI, USA, 19–22 February, 2024.
27. Wong, C.K.; Gouda, M.; Lam, S.S. Secure Group Communications Using Key Graphs. *IEEE/ACM Trans. Netw.* **2000**, *8*, 16–30. [\[CrossRef\]](#)
28. Naor, M.; Naor, D.; Lotspiech, J. Revocation and Tracing Schemes for Stateless Receivers. In Proceedings of the 21st Annual International Cryptology Conference, Santa Barbara, CA, USA, 19–23 August 2001; pp. 41–62.
29. Kaufman, C.; Hoffman, P.; Nir, Y.; Eronen, P.; Kivinen, T. RFC 7296: *Internet Key Exchange Protocol Version 2 (IKEv2)*; RFC Editor: USA, 2014. Available online: <https://datatracker.ietf.org/doc/html/rfc7296> (accessed on 22 December 2025).
30. Iyengar, J.; Thomson, M. RFC 9000: *QUIC: A UDP-Based Multiplexed and Secure Transport*; RFC Editor: USA, 2021. Available online: <https://dl.acm.org/doi/10.17487/RFC9000> (accessed on 22 December 2025).
31. Nir, Y.; Langley, A. RFC 8439: *ChaCha20 and Poly1305 for IETF Protocols*; RFC Editor: USA, 2018. Available online: <https://datatracker.ietf.org/doc/rfc8439/> (accessed on 22 December 2025).
32. Dworkin, M. Recommendation for block cipher modes of operation. *NIST Spec. Publ.* **2001**, *800*, 38B.
33. Krawczyk, H.; Eronen, P. RFC 5869: *HMAC-based Extract-and-Expand Key Derivation Function (HKDF)*; RFC Editor: USA, 2010. Available online: <https://datatracker.ietf.org/doc/html/rfc5869> (accessed on 22 December 2025).
34. Langley, A.; Hamburg, M.; Turner, S. RFC 7748: *Elliptic Curves for Security*; RFC Editor: USA, 2016. Available online: <https://www.rfc-editor.org/rfc/rfc7748.html> (accessed on 22 December 2025).
35. Biryukov, A.; Dinu, D.; Khovratovich, D.; Josefsson, S. RFC 9106: *Argon2 Memory-Hard Function for Password Hashing and Proof-of-Work Applications*; RFC Editor: USA, 2021. Available online: <https://datatracker.ietf.org/doc/rfc9106/> (accessed on 22 December 2025).
36. Biryukov, A.; Dinu, D.; Khovratovich, D. Argon2: New generation of memory-hard functions for password hashing and other applications. In Proceedings of the 2016 IEEE European Symposium on Security and Privacy (EuroS&P), Saarbruecken, Germany, 21–24 March 2016; pp. 292–302.

37. Bourdrez, D.; Krawczyk, H.; Lewi, K.; Wood, C.A. *RFC 9807: The OPAQUE Augmented Password-Authenticated Key Exchange (aPAKE) Protocol*; RFC Editor: USA, 2025. Available online: <https://datatracker.ietf.org/doc/html/rfc9807> (accessed on 22 December 2025).
38. Jarecki, S.; Krawczyk, H.; Xu, J. OPAQUE: An asymmetric PAKE protocol secure against pre-computation attacks. In Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, 29 April–3 May 2018; pp. 456–486.
39. Canetti, R.; Krawczyk, H. Analysis of key-exchange protocols and their use for building secure channels. In Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques, Innsbruck, Austria, 6–10 May 2001; Springer: Berlin/Heidelberg, Germany, 2001; pp. 453–474.
40. LaMacchia, B.; Lauter, K.; Mityagin, A. Stronger security of authenticated key exchange. In Proceedings of the International Conference on Provable Security, Wollongong, Australia, 1–2 November 2007; Springer: Berlin/Heidelberg, Germany, 2007; pp. 1–16.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.