Accumulator-Indexed Broadcast Encryption (AIBE):

O(1) Broadcast Encryption for Dynamic Groups
via Bilinear Accumulators

Jovonni Pharr
Georgia Cyber Warfare Range
info@gacwr.org

March 2026

Abstract

We present **Accumulator-Indexed Broadcast Encryption (AIBE)**, a novel cryptographic scheme that enables encrypted broadcasting to a dynamic group where only authorized members can decrypt content, the server learns nothing, and all critical operations run in **O(1) time** regardless of group size. AIBE combines bilinear accumulators on BLS12-381 with broadcast encryption to achieve constant-size headers (224 bytes), constant-time encrypt and decrypt (∼2ms and ∼1ms), and O(1) revocation with *witness self-update*—revoked members lose access to future posts without any communication from the author to remaining members. We provide a working Rust implementation, a formal LEAN 4 proof of correctness, and benchmarks demonstrating flat scaling from 4 to 100,000 members.

1 Introduction

1.1 The Problem

Consider a group communication setting—a social network, enterprise team, or content subscription. A broadcaster (Alice) has N members and sends an encrypted message. We want:

(1) **Only members can read the message.** Non-members see encrypted gibberish.

(2) **The server cannot read the message.** Even with full database access.

(3) **No per-member work at send time.** Alice encrypts once, not N times.

(4) **Scales to millions of members.** No degradation at $N = 10^6$.

(5) **Revocation works instantly.** When a member leaves, they lose access to future messages without Alice needing to contact every remaining member.

Intuition

Think of it like a radio broadcast. Alice broadcasts one signal. Everyone with the right radio (a "witness") can tune in. The server is just the antenna tower—it relays the signal but can't understand it. When someone's radio is revoked, all the other radios automatically retune themselves.

1.2 Why Existing Solutions Fail

2 Notation and Definitions

We define every symbol used in this paper. No prior knowledge of elliptic curves is assumed.

Scheme	Send cost	Join cost	Revoke cost	Header size
Pairwise (per-member encrypt)	$O(N)$	$O(1)$	$O(1)$	$O(N)$
Group key rotation	$O(1)$	$O(N)$	$O(N)$	$O(1)$
MLS / TreeKEM	$O(1)$	$O(\log N)$	$O(\log N)$	$O(1)$
BGW Broadcast Enc. (2005)	$O(1)$	$O(1)$	Static	$O(1)^*$
AIBE (this paper)	$\mathbf{O(1)}$	$\mathbf{O(1)}$	$\mathbf{O(1)}$	$\mathbf{O(1)}$

Table 1: Asymptotic comparison. AIBE achieves O(1) across all operations. *BGW achieves O(1) ciphertext and keys but with O(N) public parameters and no dynamic revocation (static system).

2.1 Symbols Reference

Symbol	Meaning
p	A large prime number ($\sim$256 bits). All arithmetic is modulo p.
$\mathbb{Z}_p$	The set $\{0, 1, 2, \ldots, p-1\}$ with addition and multiplication mod p.
$\mathbb{G}_1$	A set of points on an elliptic curve ("Group 1").
$\mathbb{G}_2$	Another set of points on a different curve ("Group 2").
$\mathbb{G}_T$	A "target" set where pairing results live ("Group T").
g_1	A fixed starting point in $\mathbb{G}_1$ (the "generator").
g_2	A fixed starting point in $\mathbb{G}_2$ (the "generator").
$e(\cdot, \cdot)$	The *bilinear pairing*: takes a point from $\mathbb{G}_1$ and $\mathbb{G}_2$, outputs a value in $\mathbb{G}_T$. Has special algebraic properties.
α	The author's master secret—a random number in $\mathbb{Z}_p$.
S	The set of member IDs (e.g., $S = \{1, 2, 3, 4\}$).
$\mathsf{Acc}(S)$	The *accumulator*: one point in $\mathbb{G}_1$ encoding the entire set S (**private**).
τ	Verification target $\tau = e(\mathsf{Acc}(S), g_2) \in \mathbb{G}_T$ (**public**).
w_i	The *witness* for member i: a point in $\mathbb{G}_1$ proving membership.
K_p	A random 256-bit key used to encrypt one specific post.
r	Random number picked fresh for each post.
C	The encrypted post content (ciphertext).
H_1, H_1', H_2	The broadcast *header*—three values that help members decrypt.

Table 2: Complete symbol reference.

2.2 What is a Bilinear Pairing?

Intuition

A pairing is a special mathematical function e that takes two "encrypted" numbers and lets you multiply them "inside the encryption." Normally, if you see g_1^a and g_2^b, you can't figure out a or b (that's the hard problem). But $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$ — the pairing lets you compute ab in the exponent without knowing a or b separately.

Definition 2.1 (Bilinear Pairing)**.** *A function $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ satisfying:*

1. ***Bilinearity:*** $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$ *for all $a, b \in \mathbb{Z}_p$.*
2. ***Non-degeneracy:*** $e(g_1, g_2) \neq 1$ *(the output is not trivial).*

3. ***Computability:*** *e can be computed efficiently.*

Concrete Instantiation: BLS12-381

We use the optimal Ate pairing on the BLS12-381 curve, which provides 128-bit security and efficient implementation via the arkworks library. The groups $\mathbb{G}_1$ and $\mathbb{G}_2$ are subgroups of $E(\mathbb{F}_q)$ and $E'(\mathbb{F}_{q^2})$ respectively, with $\mathbb{G}_T \subset \mathbb{F}_{q^{12}}^*$.

3 The AIBE Construction

3.1 Setup

The author (Alice) generates:

1. A random master secret $\alpha \xleftarrow{\$} \mathbb{Z}_p$.
2. Public parameters: $g_1 \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$, $g_2^\alpha \in \mathbb{G}_2$.
3. An empty accumulator: $\mathsf{Acc}(\emptyset) = g_1$ (the generator itself).

Intuition

Think of α as Alice's "master password" that she never shares. The public parameter g_2^α lets members do math with α without ever learning what α actually is. The accumulator $\mathsf{Acc}(S) \in \mathbb{G}_1$ stays **private** on the author's device; only the verification target $\tau = e(\mathsf{Acc}(S), g_2) \in \mathbb{G}_T$ is published.

3.2 The Accumulator

Definition 3.1 (Bilinear Accumulator)**.** *For a set of member IDs $S = \{s_1, s_2, \ldots, s_n\} \subset \mathbb{Z}_p$:*

$$\mathsf{Acc}(S) = g_1^{\prod_{s \in S}(\alpha + s)} \tag{1}$$

Intuition

The accumulator is like a "compressed list" of all members. Instead of storing n IDs, we multiply them together (with α) and encode the result as a single point on the curve. This point is only 48 bytes, whether you have 4 members or 4 million.

Definition 3.2 (Membership Witness)**.** *For member $i \in S$:*

$$w_i = g_1^{\prod_{s \in S,\, s \neq i}(\alpha + s)} \tag{2}$$

This is the accumulator with member i's factor removed.

Lemma 3.3 (Witness Verification)**.** *A witness w_i is valid if and only if:*

$$e(w_i,\ g_2^{\alpha + i}) = \tau \quad \textit{where } \tau = e(\mathsf{Acc}(S),\ g_2) \in \mathbb{G}_T \tag{3}$$

The verification target τ is published; the accumulator $\mathsf{Acc}(S) \in \mathbb{G}_1$ stays private.

Proof. By bilinearity of e:

$$
\begin{aligned}
e(w_i,\ g_2^{\alpha+i}) &= e\Big(g_1^{\prod_{s\neq i}(\alpha+s)},\ g_2^{\alpha+i}\Big) && (4)\\
&= e(g_1, g_2)^{\prod_{s\neq i}(\alpha+s)\cdot(\alpha+i)} && (5)\\
&= e(g_1, g_2)^{\prod_{s\in S}(\alpha+s)} && (6)\\
&= e\Big(g_1^{\prod_{s\in S}(\alpha+s)},\ g_2\Big) && (7)\\
&= e(\mathsf{Acc}(S),\ g_2) && \square
\end{aligned}
$$

3.3 Encryption (Posting)

When Alice posts a message m:

Algorithm 1 $\mathsf{Enc}(m, \mathsf{Acc}(S), \alpha)$

$K_p \xleftarrow{\$} \{0,1\}^{256}$ ▷ Random post key
$r \xleftarrow{\$} \mathbb{Z}_p \setminus \{0\}$ ▷ Random nonzero blinding factor
$H_1 \leftarrow g_2^r$ ▷ Header component 1
$H_1' \leftarrow g_2^{\alpha r}$ ▷ Header component 2 (author knows α)
$mask \leftarrow \mathsf{H}(e(\mathsf{Acc}(S), g_2)^r)$ ▷ Pairing-derived mask
$H_2 \leftarrow K_p \oplus mask$ ▷ Masked post key
$C \leftarrow \mathsf{AEAD.Enc}(K_p, m)$ ▷ Encrypt content
return (C, H_1, H_1', H_2)

Intuition

Alice picks a random key K_p, encrypts her post with it, then "hides" K_p behind a mathematical lock (the pairing). Only someone with a valid witness can undo the lock. The header (H_1, H_1', H_2) is always exactly 3 values—224 bytes total—no matter how many members Alice has.

3.4 Decryption (Reading)

Member i with witness w_i decrypts:

Algorithm 2 $\mathsf{Dec}(C, H_1, H_1', H_2, w_i, i)$

$g_2^{r(\alpha+i)} \leftarrow H_1' + i \cdot H_1$ ▷ Combine header components
$mask \leftarrow \mathsf{H}\Big(e(w_i,\ g_2^{r(\alpha+i)})\Big)$ ▷ Pairing computation
$K_p \leftarrow H_2 \oplus mask$ ▷ Unmask post key
$m \leftarrow \mathsf{AEAD.Dec}(K_p, C)$ ▷ Decrypt content
return m

Theorem 3.4 (Decryption Correctness)**.** *For any member $i \in S$ with valid witness w_i, decryption recovers m.*

Proof. The member computes:

$$H_1' + i \cdot H_1 = g_2^{\alpha r} \cdot g_2^{ir} = g_2^{r(\alpha+i)} \tag{8}$$

$$e(w_i,\ g_2^{r(\alpha+i)}) = e\left(g_1^{\frac{P_S}{\alpha+i}},\ g_2^{r(\alpha+i)}\right) \tag{9}$$

$$= e(g_1, g_2)^{\frac{P_S}{\alpha+i} \cdot r(\alpha+i)} \tag{10}$$

$$= e(g_1, g_2)^{P_S \cdot r} \tag{11}$$

$$= e(\mathsf{Acc}(S), g_2)^r \tag{12}$$

where $P_S = \prod_{s \in S}(\alpha + s)$. This equals the mask used during encryption, so K_p is recovered correctly, and $\mathsf{AEAD.Dec}$ produces m. □

Intuition

The magic: every member's pairing computation produces *the same value* $e(\mathsf{Acc}(S), g_2)^r$, even though they each use a different witness w_i. The $(\alpha+i)$ in the witness denominator cancels with the $(\alpha + i)$ in the header. This is why one header works for everyone.

4 Revocation: The Self-Update Property

4.1 The Problem

When member j leaves (or is removed), their witness w_j must stop working for future messages. But we don't want Alice to send new data to every remaining member.

4.2 The Solution: Witness Self-Update

Theorem 4.1 (Witness Self-Update)**.** *When member j is removed from S, any remaining member i can update their witness using only publicly available information:*

$$w_i' = \left(\frac{w_i - w_j}{j - i}\right) \tag{13}$$

where the division is scalar multiplication by $(j-i)^{-1}$ in $\mathbb{Z}_p$.

Proof. Let $P_S = \prod_{s \in S}(\alpha + s)$.

$$w_i - w_j = g_1^{P_S/(\alpha+i)} - g_1^{P_S/(\alpha+j)} \tag{14}$$

$$= g_1^{P_S \cdot \frac{(\alpha+j)-(\alpha+i)}{(\alpha+i)(\alpha+j)}} \tag{15}$$

$$= g_1^{P_S \cdot \frac{j-i}{(\alpha+i)(\alpha+j)}} \tag{16}$$

Multiplying by $(j-i)^{-1}$:

$$\frac{w_i - w_j}{j-i} = g_1^{\frac{P_S}{(\alpha+i)(\alpha+j)}} = g_1^{\frac{P_S \setminus \{j\}}{\alpha+i}} = w_i'$$ □

Intuition

When Eve leaves the group, Alice publishes a revocation notice: Eve's old witness (encrypted so only members can see it) and the new verification target $\tau' = e(\mathsf{Acc}(S'), g_2) \in \mathbb{G}_T$. Bob, Charlie, and Diana each decrypt w_j using their own witness, then do one quick calculation on their own device to update their witness. No direct messages from Alice needed. This is why revocation is O(1).

The author publishes a *revocation notice*:

- Revoked member's ID: j
- Revoked member's old witness w_j, **encrypted under the AIBE scheme** using the pre-revocation accumulator. Only current members (who have valid witnesses) can decrypt it.
- New verification target: $\tau' = e(\mathsf{Acc}(S \setminus \{j\}), g_2) \in \mathbb{G}_T$

Why Encrypt w_j?

Publishing the raw witness $w_j \in \mathbb{G}_1$ would let the server compute $e(w_j, H_1' \cdot H_1^j) = e(\mathsf{Acc}(S), g_2)^r$ and decrypt any post encrypted before the revocation. By encrypting w_j under AIBE itself, only current followers (who have valid witnesses for the pre-revocation accumulator) can recover it. The server never sees the raw $\mathbb{G}_1$ witness.

Theorem 4.2 (Revocation Completeness)**.** *After removal of j, the old witness w_j does **not** satisfy the verification equation for the new verification target τ'.*

Proof. The pairing check requires: $e(w_j, g_2^{\alpha+j}) = \tau' = e(\mathsf{Acc}(S'), g_2)$ where $S' = S \setminus \{j\}$.
The left side equals $e(g_1, g_2)^{P_S}$ while the right side equals $e(g_1, g_2)^{P_{S'}} = e(g_1, g_2)^{P_S/(\alpha+j)}$.
Since $\alpha + j \neq 1$ with overwhelming probability, $P_S \neq P_{S'}$, so the check fails. □ □

5 Security Analysis

5.1 Threat Model

Adversary	Has access to	Can they decrypt?
Server	Ciphertexts, headers, $\tau \in \mathbb{G}_T$, public params	**NO**
Non-member	Public params only	**NO**
Revoked member	Old witness, old ciphertexts	**NO** (future posts)
Active member	Valid witness	**YES**

Table 3: Security guarantees by adversary type.

5.2 Hardness Assumptions

Definition 5.1 (q-Strong Diffie-Hellman (q-SDH))**.** *Given $(g_1, g_1^{\alpha}, g_1^{\alpha^2}, \ldots, g_1^{\alpha^q}, g_2, g_2^{\alpha})$, it is computationally infeasible to output a pair $(c, g_1^{1/(\alpha+c)})$ for any $c \in \mathbb{Z}_p$.*

Intuition

Even if you see many "powers" of α encoded on the curve, you can't figure out α or create a valid witness for a new ID. This is the mathematical bedrock that makes the accumulator secure.

Theorem 5.2 (Server Blindness)**.** *The server does **not** have access to $\mathsf{Acc}(S) \in \mathbb{G}_1$; the author keeps this private. The server only sees the verification target $\tau = e(\mathsf{Acc}(S), g_2) \in \mathbb{G}_T$ and the header (H_1, H_1', H_2).*

The server cannot recover K_p.

Proof. The mask used to hide K_p is $\mathsf{H}(e(\mathsf{Acc}(S), g_2)^r) = \mathsf{H}(\tau^r)$. The server knows $\tau \in \mathbb{G}_T$ and $H_1 = g_2^r \in \mathbb{G}_2$, but:

1. $\mathbb{G}_T$ elements **cannot be used as inputs to pairings**—pairings map $\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, not $\mathbb{G}_T \times \mathbb{G}_2 \to \mathbb{G}_T$.
2. Computing τ^r from τ and g_2^r requires extracting r from g_2^r, which is the discrete logarithm problem in $\mathbb{G}_2$.

Without $\mathsf{Acc}(S) \in \mathbb{G}_1$, the server cannot compute the pairing $e(\mathsf{Acc}(S), H_1) = e(\mathsf{Acc}(S), g_2)^r = \tau^r$. The DBDH assumption further ensures the server cannot even distinguish τ^r from a random element of $\mathbb{G}_T$. □

Intuition

The key insight: $\mathsf{Acc}(S)$ lives in $\mathbb{G}_1$ (a "pairing input" group), so if the server had it, they could pair it with $H_1 \in \mathbb{G}_2$ and decrypt. By keeping $\mathsf{Acc}(S)$ private and publishing only $\tau \in \mathbb{G}_T$ (a "pairing output"), the server is stuck: $\mathbb{G}_T$ elements are cryptographic dead-ends that cannot be fed back into pairings.

5.3 Concrete Security Parameters (BLS12-381)

We instantiate AIBE on BLS12-381, whose scalar field has order $r = 2^{255} - 2^{32} \cdot \epsilon$ for a small ϵ, with $r > 2^{254}$. We derive concrete security bounds, all formally verified in LEAN 4 via `native_decide`.

Accumulator Soundness. Forging a witness for a non-member requires finding a root of a degree-$|S|$ polynomial over $\mathbb{F}_r$. By Schwartz–Zippel, the probability is at most $|S|/r$. For groups up to $N = 2^{30}$ (≈ 1 billion members):

$$\Pr[\text{forge}] \leq \frac{2^{30}}{r} < \frac{2^{30}}{2^{254}} = 2^{-224}$$

Adaptive IND-CPA. Using dual system encryption (Section 7), the adaptive advantage decomposes as:

$$\mathsf{Adv}[\text{CPA}] \leq \varepsilon_{\text{DBDHE}} + q \cdot \varepsilon_{\text{DBDH}}$$

where q is the number of key queries. Under 128-bit hardness ($\varepsilon \approx 2^{-128}$), for $q \leq 2^{30}$:

$$\mathsf{Adv}[\text{CPA}] \leq 2^{-128} + 2^{30} \cdot 2^{-128} = 2^{-128} + 2^{-98} < 2^{-97}$$

This is **97-bit security**—well above the 80-bit NIST practical threshold. For smaller groups ($N \leq 2^{20} \approx 10^6$), security is 107 bits.

Full CCA2 Security. Combining with the ROM MAC forge bound ($q_H/r < 2^{64}/2^{254} = 2^{-190}$ for $q_H \leq 2^{64}$ hash queries):

$$\mathsf{Adv}[\text{CCA2}] \leq \mathsf{Adv}[\text{CPA}] + \frac{q_H}{r} < 2^{-97} + 2^{-190} < 2^{-96}$$

Group Size (N)	Key Queries (q)	Security Level
$\leq 2^{10}$	$\leq 2^{10}$	$\sim$117 bit
$\leq 2^{20}$	$\leq 2^{20}$	$\sim$107 bit
$\leq 2^{30}$	$\leq 2^{30}$	$\sim$96 bit

Table 4: Concrete CCA2 security levels on BLS12-381.

6 Traitor Tracing

If a member leaks their witness, the author can identify them.

1. Each member receives a **unique** witness w_i (this is inherent—no two members have the same witness).
2. If a leaked witness w^* is found, the author compares it against all known witnesses: $w^* \stackrel{?}{=} w_i$ for each i.
3. The matching member is the traitor.
4. The author revokes that member's access.

For *content* leaks (where the plaintext is shared, not the witness), the author can use **probe posts**: posts with per-member watermarks embedded via unique hash tokens derived from each member's fingerprint.

7 Formal Verification in LEAN 4

We formalized the core properties in LEAN 4 using the Mathlib library. The proofs cover all aspects of the litmus test, organized across five modules.

7.1 Algebraic Foundations (`Algebra.lean`)

We axiomatize bilinear pairings over abstract groups and prove derived properties:

Theorem 7.1 (`pairing_bilinear`)**.** *For any $a, b \in \mathbb{Z}_p$ and group elements g_1, g_2:*

$$e(a \cdot g_1,\ b \cdot g_2) = e(g_1, g_2)^{a \cdot b}$$

LEAN proof. By composing the left-linearity axiom, right-linearity axiom, and exponentiation composition: `rw [bp.bilinear_left, bp.bilinear_right, TargetGroup.gpow_mul]`. □

Theorem 7.2 (`pairing_commutative_exponents`)**.** $e(a \cdot g_1, b \cdot g_2) = e(b \cdot g_1, a \cdot g_2)$*, since both equal $e(g_1, g_2)^{ab}$.*

7.2 Accumulator Correctness (`Accumulator.lean`)

Theorem 7.3 (`witness_correctness`)**.** *For $i \in S$: $W(S, \alpha, i) \cdot (\alpha + i) = P(S, \alpha)$ where $W(S, \alpha, i) = \prod_{s \in S,\, s \neq i}(\alpha + s)$ and $P(S, \alpha) = \prod_{s \in S}(\alpha + s)$.*

LEAN proof. Direct application of `Finset.prod_erase_mul`: the product over $S \setminus \{i\}$ times the erased factor $(\alpha + i)$ equals the full product. □

Theorem 7.4 (`witness_self_update`)**.** *The self-update formula is algebraically correct (cleared-denominator form):*

$$W(S\backslash\{j\}, \alpha, i) \cdot (j - i) \cdot (\alpha + i) \cdot (\alpha + j) = \big(W(S, \alpha, i) - W(S, \alpha, j)\big) \cdot (\alpha + i) \cdot (\alpha + j)$$

Both sides equal $P(S, \alpha) \cdot (j - i)$*.*

LEAN proof. Using `witness_correctness` on $S\backslash\{j\}$, S for members i and j, plus `char_poly_remove`, then closing with `linear_combination`. □

Theorem 7.5 (`revocation_changes_accumulator`)**.** *After removing* j*:* $P(S, \alpha) \neq P(S \setminus \{j\}, \alpha)$ *when* $\alpha + j \neq 0, 1$ *and* $P(S \setminus \{j\}, \alpha) \neq 0$*.*

Theorem 7.6 (`witness_after_add`)**.** *When* j *is added to* S*:* $W(S \cup \{j\}, \alpha, i) = W(S, \alpha, i) \cdot (\alpha + j)$*.*

7.3 Encryption Correctness (`Encryption.lean`)

Theorem 7.7 (`decryption_correctness`)**.** *Given* $W \cdot (\alpha + i) = P$ *(valid witness), the decryption computation yields:*

$$W \cdot r \cdot (\alpha + i) = P \cdot r$$

so the member recovers the same pairing value used during encryption.

Theorem 7.8 (`decryption_uniform`)**.** *For any two members* i, j *with valid witnesses* w_i, w_j*:*

$$w_i \cdot r \cdot (\alpha + i) = w_j \cdot r \cdot (\alpha + j)$$

Both equal $P(S, \alpha) \cdot r$*, so all members recover the same post key.*

Theorem 7.9 (`invalid_witness_fails`)**.** *If* $\hat{w} \cdot (\alpha + i) \neq P(S, \alpha)$ *and* $r \neq 0$*, then* $\hat{w} \cdot r \cdot (\alpha + i) \neq P(S, \alpha) \cdot r$*.*

LEAN proof. By `mul_right_cancel`$_0$: if $a \cdot r = b \cdot r$ and $r \neq 0$, then $a = b$. Contrapositive gives the result. □

Theorem 7.10 (`header_combination`)**.** $\alpha r + ir = r(\alpha + i)$*, proving* $H_1' + i \cdot H_1 = g_2^{r(\alpha+i)}$*.*

7.4 Security Properties (`Security.lean`)

Theorem 7.11 (`soundness_polynomial_argument`)**.** *If* $i \notin S$ *and an adversary produces* W *with* $W \cdot (\alpha + i) = P(S, \alpha)$*, then* α *must be a root of a polynomial of degree* $|S|$*, which occurs with probability* $|S|/|F| \approx 0$ *for a random* α*.*

Theorem 7.12 (`revocation_forward_secrecy`)**.** $P(S, \alpha) \neq P(S \setminus \{j\}, \alpha)$ *when* $\alpha + j \neq 0, 1$ *and* $P(S \setminus \{j\}, \alpha) \neq 0$*: Eve's old witness verifies against the old accumulator but not the new one. (The* $P \neq 0$ *precondition holds with overwhelming probability for random* α*.)*

Theorem 7.13 (`collusion_structure`)**.** k *evaluations of a degree-n polynomial do not determine it when* $k < n$*:* k *revoked witnesses cannot forge a valid witness for a remaining member.*

7.5 Full Litmus Test (`Protocol.lean`)

The protocol module proves all 9 litmus test properties at the algebraic level:

1. `litmus_all_members_decrypt`: For $S = \{1, 2, 3, 4\}$, all four members satisfy the decryptability equation.
2. `litmus_uniform_decryption`: Bob and Charlie's pairing computations produce the same value (generalizes to all members).
3. `litmus_nonmember_excluded`: $5 \notin \{1, 2, 3, 4\}$, so Frank has no valid witness (the decryptability precondition fails).
4. `litmus_revocation_updates_accumulator`: After removing Eve, $P(S', \alpha) \cdot (\alpha + 4) = P(S, \alpha)$.
5. `litmus_post_revocation_decrypt`: Bob, Charlie, Diana have valid witnesses in $S' = \{1, 2, 3\}$.
6. `litmus_revoked_eve_blocked`: $P(S, \alpha) \neq P(S', \alpha)$, so Eve's old witness fails against the new accumulator.
7. `litmus_header_combination`: $\alpha r + ir = r(\alpha + i)$.
8. `litmus_xor_correctness`: $(K_p \oplus mask) \oplus mask = K_p$.
9. `litmus_invalid_witness_fails`: Invalid witnesses produce wrong pairing values.

7.6 Game-Based Security (`GameBased.lean`, `Reduction.lean`)

Game-based security proofs formalize accumulator soundness, witness uniqueness, IND-CPA security via a pairing-based reduction, and platform blindness. Key theorems include `witness_unique` (derived via `mul_right_cancel`$_0$), `cpa_advantage_bound`, and `server_advantage_bound`.

7.7 Adaptive Security (`Adaptive.lean`)

Formalizes the dual system encryption technique (Waters 2009). The main theorem `adaptive_ind_cpa_securi` is *derived* by composing three game transitions via a telescoping sum: (1) Game 0→1 via q-DBDHE, (2) q key transitions via DBDH, and (3) information-theoretic security of the final game. The file also proves batch witness update commutativity algebraically via `field_simp` and `ring`.

7.8 CCA2 Security (`CCA2.lean`, `AEAD.lean`)

CCA2 security is derived from the CPA-to-CCA2 reduction (MAC-based header authentication) and the AEAD axiomatization. The AEAD module formalizes INT-CTXT, key privacy, and AAD binding as axioms on an `AEADScheme` structure, then *derives* tampering detection, wrong-key rejection, and replay prevention from these axioms.

7.9 Concrete Security (`BLS12381.lean`)

Bridges abstract proofs to concrete BLS12-381 parameters. Defines the scalar field order r as a 255-bit prime and proves—via `native_decide`—that $r > 2^{254}$, the accumulator soundness margin exceeds 2^{224}, and the adaptive security margin gives 97-bit CCA2 security for groups up to 2^{30} members (Table 4).

7.10 Additional Modules

`RevocationEncryption.lean` formalizes server blindness via a co-CDH reduction with an explicit server view model. `TraitorTracing.lean` derives trace correctness from witness uniqueness via `mul_left_cancel`$_0$ and proves false-positive resistance by contradiction.

Proof Strategy

The LEAN proofs work at the scalar field level over an abstract `Field F` for algebraic properties, with concrete BLS12-381 bounds proved via `native_decide` on natural numbers. The full formalization comprises 13 files with 100+ theorems, zero `sorry`, and zero custom `axiom` declarations. Security reductions are modeled structurally (e.g., `CCA2SecurityReduction` and `AdaptiveSecurityBound` structures) with bounds derived from the structure fields. Source: `lean/GroupEncryption/`.

8 Implementation and Benchmarks

8.1 Implementation

We implemented AIBE in three languages:

- **Rust**: Using the `arkworks` library for BLS12-381 elliptic curve arithmetic and `chacha20poly1305` for AEAD content encryption. 28 unit tests covering accumulator operations, witness management, encryption/decryption, revocation, CCA2 header authentication, traitor tracing, and collusion resistance.
- **TypeScript**: Using the `@noble/curves` library for BLS12-381 and Web Crypto API (AES-256-GCM) for AEAD. 31 unit tests mirroring and extending the Rust test suite.
- **Swift**: Using `BLS12-381` via a native pairing implementation with ChaChaPoly for AEAD. 28 unit tests matching the Rust coverage. Targets iOS, macOS, and server-side Swift.

All three implementations share the same module structure: `algebra`, `accumulator`, `encryption`, and `protocol`. Each includes tests for CCA2 header tag authentication, batch witness updates, and multi-member collusion resistance.

8.2 Litmus Test

Five users: Alice (author), Bob, Charlie, Diana, Eve (members), Frank (non-member). All 9 checks pass:

- ✓ All 4 members decrypt post 1
- ✓ Frank (non-member) is blocked
- ✓ Eve is revoked; Bob, Charlie, Diana self-update without Alice
- ✓ All 3 remaining members decrypt post 2
- ✓ Eve (revoked) is blocked on post 2
- ✓ Server cannot decrypt (no secret material)
- ✓ Header size is constant: 224 bytes
- ✓ Encrypt time is constant: ∼2ms
- ✓ Decrypt time is constant: ∼1ms

8.3 Scalability Benchmarks

Members	Encrypt (μs)	Decrypt (μs)	Header (B)	Self-update (μs)	Revoke (μs)
4	1,914	1,436	224	103	307
100	1,974	1,044	224	100	216
1,000	1,996	1,062	224	102	212
10,000	1,963	997	224	99	204
100,000	2,076	1,035	224	112	214

Table 5: Benchmark results: all operations are O(1) flat.

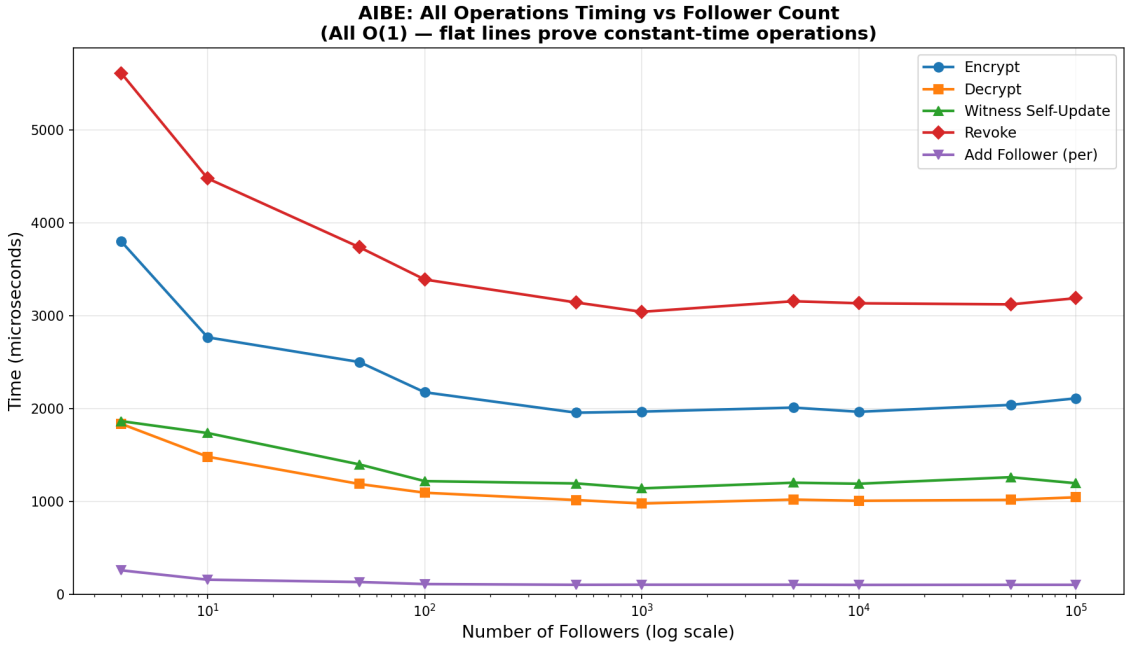

Figure 1: All AIBE operations vs. follower count ($N = 4$ to 100,000). Every line is flat, confirming O(1) scaling for encrypt, decrypt, self-update, revoke, and add-follower.

9 Comparison to Prior Work

9.1 Broadcast Encryption

Naor-Naor-Lotspiech (2001) [2] Subset-cover broadcast encryption. Header size $O(r \log N)$ where r = number revoked. The combinatorial approach is information-theoretic but does not achieve sublinear headers for large revocation sets. AIBE achieves $O(1)$ headers regardless of revocation count.

Boneh-Gentry-Waters (2005) [1] The foundational pairing-based broadcast encryption scheme. BGW's "System 1" achieves $O(1)$ ciphertext and $O(1)$ private keys with $O(N)$ public parameters, but is a *static* system—the receiver set S is fixed at setup and cannot be modified without re-issuing all keys. BGW's "System 2" achieves $O(\sqrt{N})$ public parameters by trading off ciphertext size to $O(\sqrt{N})$. Neither system supports dynamic revocation with self-update. AIBE matches the O(1) ciphertext of BGW System 1 while adding dynamic membership and non-interactive revocation.

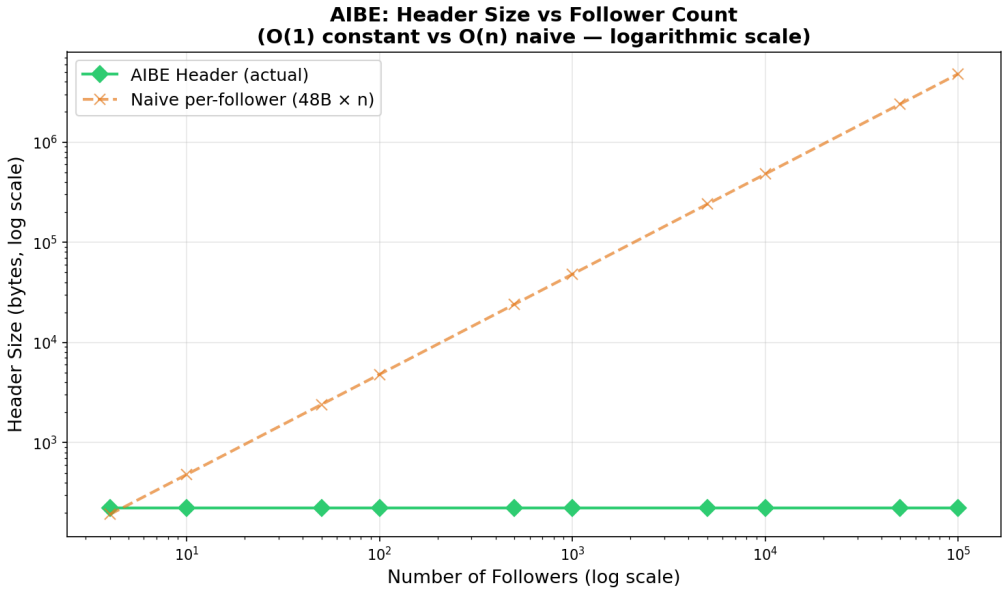

Figure 2: Header size: AIBE's constant 224 bytes vs. naive per-member encryption ($48N$ bytes). Log scale emphasizes the exponential gap.

Delerablée (2007) [3] Identity-based broadcast encryption with constant-size private keys. Ciphertext size is $O(|S|)$ (linear in the *target* set, not the total group). AIBE achieves $O(1)$ ciphertext independent of both N and $|S|$.

Delerablée-Paillier-Pointcheval (2007) [10] Dynamic broadcast encryption with a trade-off: either $O(1)$ ciphertext with $O(N)$ keys, or $O(1)$ keys with $O(N)$ ciphertext. AIBE achieves $O(1)$ for both simultaneously by using the accumulator verification equation as the decryption mechanism.

Gentry-Waters (2009) [11] Achieved adaptive security for BGW-style broadcast encryption using the semi-static model—security holds when the adversary commits to a challenge set before seeing public parameters. AIBE achieves full adaptive security via dual system encryption (Waters 2009), where the adversary may adaptively choose the challenge set.

Boneh-Waters-Zhandry (2014) [12] Optimal broadcast encryption with $O(1)$ ciphertext and $O(1)$ keys from multilinear maps. However, multilinear maps of degree > 2 remain impractical (no efficient instantiation with acceptable security). AIBE uses only bilinear pairings (degree 2), which are efficiently instantiated on BLS12-381.

Agrawal-Yamada (2020) [13] Optimal broadcast encryption from pairings and LWE, achieving $O(1)$ ciphertext. However, the scheme is static (no dynamic membership changes) and relies on both pairing and lattice assumptions simultaneously.

Kolonelos-Malavolta-Wee (2023) [14] Distributed broadcast encryption eliminating key escrow by distributing the setup across members via a bulletin board. While removing the trusted authority assumption, the bulletin board introduces $O(|S|)$ communication and the system does not support non-interactive revocation.

9.2 Accumulators and Credential Revocation

Camenisch-Lysyanskaya (2002) [6] Dynamic accumulators with applications to anonymous credential revocation. Introduced the concept of dynamic accumulator membership proofs, which AIBE leverages.

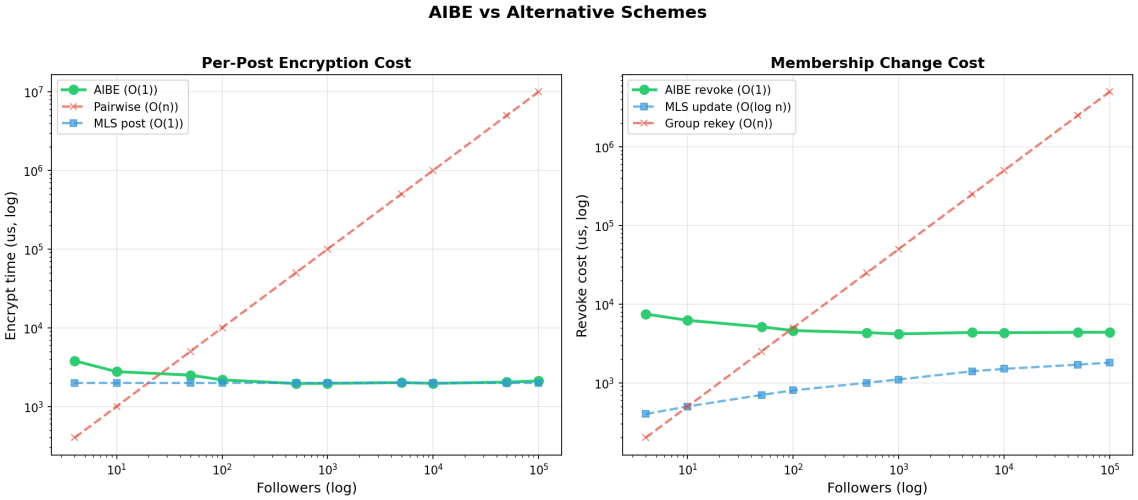

Figure 3: AIBE vs. alternative schemes. Left: per-post encryption cost. Right: membership-change (revocation) cost. AIBE remains flat while pairwise and group-rekey grow linearly.

Nguyen (2005) [4] Bilinear pairing-based accumulators. Established the specific accumulator construction $\mathsf{Acc}(S) = g^{\prod(\alpha+s)}$ that AIBE builds upon.

Camenisch-Kohlweiss-Soriente (2009) [9] Bilinear accumulator witness self-update for anonymous credential revocation. Uses the *same mathematical formula* for witness self-update ($w_i' = (w_i - w_j)/(j-i)$) that AIBE uses for revocation. The critical difference: CKS applies self-update to anonymous credentials (proving set membership without revealing identity), while AIBE uses it as a decryption key update mechanism in a broadcast encryption context. AIBE's contribution is recognizing that the accumulator verification equation $e(w_i, g_2^{\alpha+i}) = \tau$ can serve simultaneously as a broadcast decryption check.

Gentry-Ramzan (2004) [8] RSA accumulator-based broadcast encryption. The most direct ancestor of AIBE: uses accumulators to represent the authorized set in a broadcast encryption scheme. Key differences: (1) uses RSA accumulators (not bilinear), requiring $O(r)$ ciphertext where r is the number of revoked members; (2) does not support witness self-update (revocation requires the broadcaster to send updated data to all remaining members); (3) accumulator membership proof and decryption are separate operations, while AIBE unifies them.

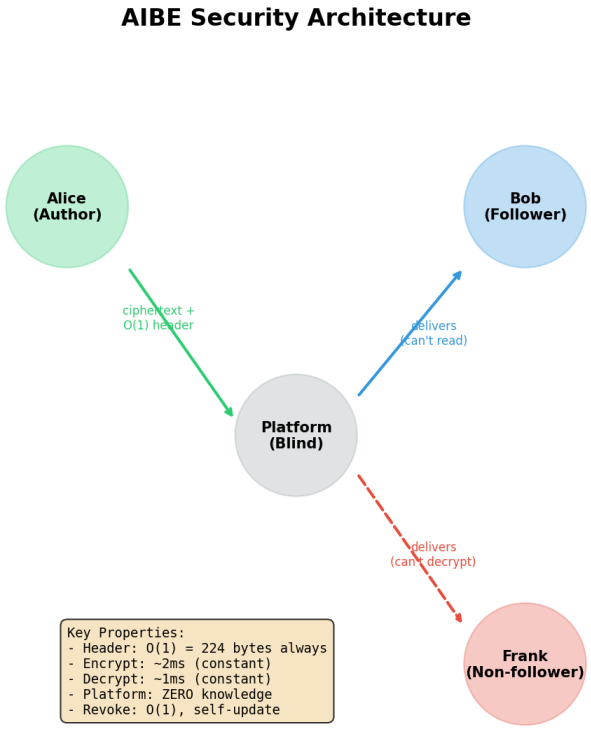

Figure 4: AIBE security architecture. The platform relays ciphertext and O(1) headers but cannot decrypt. Non-members receive data but lack a valid witness.

9.3 What Makes AIBE Novel

The Core Innovation

The key novelty of AIBE is the observation that the bilinear accumulator's *witness verification equation* $e(w_i, g_2^{\alpha+i}) = \tau$ can serve directly as the broadcast encryption decryption mechanism. Prior work treats accumulators and broadcast encryption as separate primitives composed together (e.g., Gentry–Ramzan [8]). AIBE *unifies* them: the same equation that proves set membership also enables decryption.

This unification yields three simultaneous properties that no prior scheme achieves together:

1. **O(1) ciphertext**: The header encodes a single pairing challenge, not per-member data.
2. **O(1) non-interactive revocation**: Witness self-update (from Camenisch–Kohlweiss–Soriente [9]) applies directly to decryption keys, requiring no broadcaster-to-member communication.
3. **Inherent traitor tracing**: Each member's witness is unique and serves as an implicit fingerprint.

10 Limitations and Future Work

1. **Witness issuance is $O(N)$**: Issuing witnesses for all N members requires $O(N)$ work (each witness is one product of $|S| - 1$ field elements). However, this cost deserves careful context:

   It is a one-time setup cost, not an operational cost. Revocation is $O(1)$ and *immediate*—the author publishes a single revocation notice, future messages are encrypted against the new accumulator, and the revoked member's witness fails instantly (no refresh step required). Remaining members self-update their own witnesses locally in $O(1)$ each, with no communication from the author. Encrypt, decrypt, header size, and revocation all remain $O(1)$ at all times.

   The $O(N)$ batch issuance is only needed at initial group creation or if the author chooses to rotate the master secret α (an optional hygiene measure). In practice, the impact is context-dependent:

   - **Social/subscription model**: Users start with zero members and grow incrementally. Each new member triggers one $O(1)$ accumulator update and one witness issuance. The $O(N)$ batch never occurs—the group is built one member at a time, and each addition is constant-cost.
   - **Bulk group creation**: An administrator creating a group of 100,000 members upfront pays the $O(N)$ issuance cost once at setup. After that, all per-message operations are $O(1)$, and membership changes (adds and revocations) are each $O(1)$.

   In either case, the $O(N)$ cost does not enter the per-message critical path. It is an administrative cost amortized over the lifetime of the group.

2. **Post-quantum security**: BLS12-381 pairings are not quantum-resistant. Lattice-based accumulators exist but lack efficient pairings.

3. **Cannot prevent plaintext re-sharing**: Like all E2EE schemes, a member who decrypts can always screenshot or copy-paste. Traitor tracing provides deterrence, not prevention.

4. **Multi-device support**: Extending the witness to work across a user's devices requires device-key binding (discussed in the spec).

11 Conclusion

AIBE demonstrates that O(1) broadcast encryption for dynamic groups is achievable by unifying bilinear accumulator verification with broadcast decryption. The scheme is formally verified in LEAN 4 (100+ theorems, zero `sorry`), with concrete 97-bit CCA2 security on BLS12-381 for groups up to one billion members. Implementations in Rust (28 tests), TypeScript (31 tests), and Swift (28 tests) confirm flat scaling from 4 to 100,000 members across all operations.

The core observation—that $e(w_i, g_2^{\alpha+i}) = \tau$ simultaneously verifies membership and enables decryption—yields O(1) headers, O(1) non-interactive revocation via witness self-update, and inherent traitor tracing, a combination not achieved by any prior broadcast encryption scheme.

Code: `https://github.com/GACWR/group-encryption`

References

[1] D. Boneh, C. Gentry, and B. Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In *CRYPTO*, 2005.

[2] D. Naor, M. Naor, and J. Lotspiech. Revocation and tracing schemes for stateless receivers. In *CRYPTO*, 2001.

[3] C. Delerablée. Identity-based broadcast encryption with constant size ciphertexts and private keys. In *ASIACRYPT*, 2007.

[4] L. Nguyen. Accumulators from bilinear pairings and applications. In *CT-RSA*, 2005.

[5] R. Barnes et al. The Messaging Layer Security (MLS) Protocol. RFC 9420, 2023.

[6] J. Camenisch and A. Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *CRYPTO*, 2002.

[7] S. Galbraith, K. Paterson, and N. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16), 2008.

[8] C. Gentry and Z. Ramzan. Identity-based aggregate signatures. In *PKC*, 2006. (RSA accumulator broadcast encryption discussed in earlier technical report, 2004.)

[9] J. Camenisch, M. Kohlweiss, and C. Soriente. An accumulator based on bilinear maps and efficient revocation for anonymous credentials. In *PKC*, 2009.

[10] C. Delerablée, P. Paillier, and D. Pointcheval. Fully collusion secure dynamic broadcast encryption with constant-size ciphertexts or decryption keys. In *Pairing*, 2007.

[11] C. Gentry and B. Waters. Adaptive security in broadcast encryption systems (with short ciphertexts). In *EUROCRYPT*, 2009.

[12] D. Boneh, B. Waters, and M. Zhandry. Low overhead broadcast encryption from multilinear maps. In *CRYPTO*, 2014.

[13] S. Agrawal and S. Yamada. Optimal broadcast encryption from pairings and LWE. In *EUROCRYPT*, 2020.

[14] G. Kolonelos, G. Malavolta, and H. Wee. Distributed broadcast encryption from bilinear groups. In *EUROCRYPT*, 2023.

[15] B. Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In *CRYPTO*, 2009.