Forward Secure Sequential Aggregate Authentication

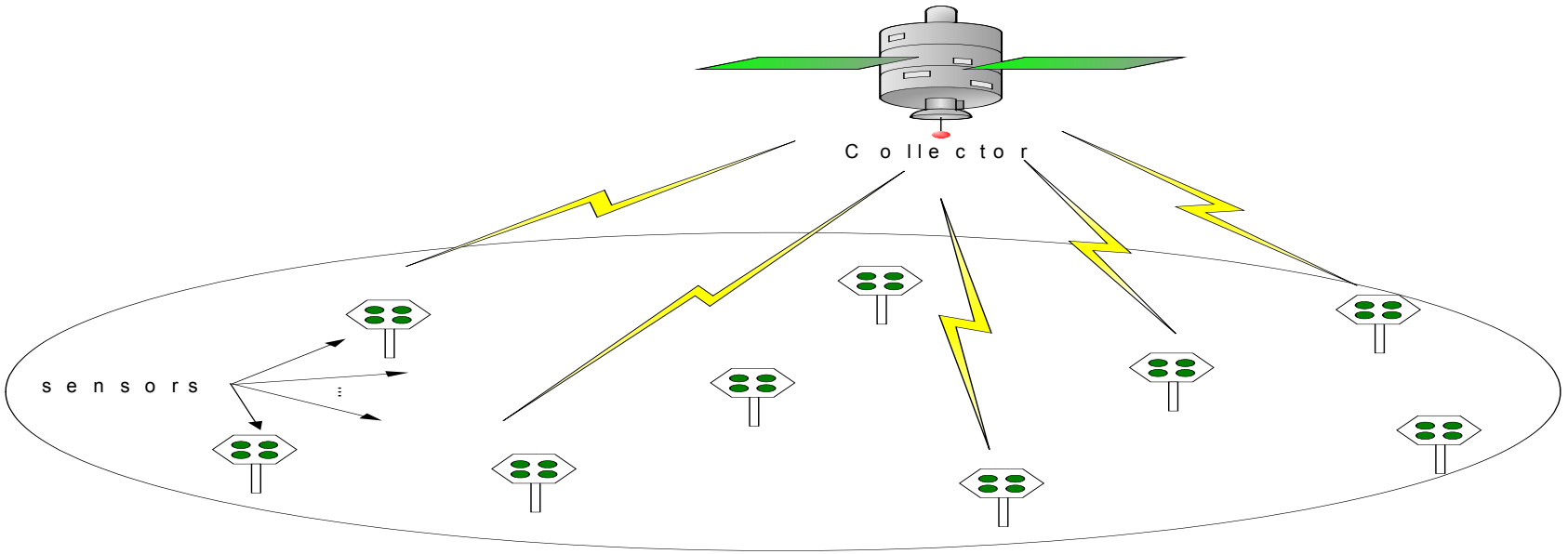
Di Ma and Gene Tsudik

Computer Science Department
University of California, Irvine

Motivation

Unattended sensors:

- Sensors do not network
- Sensors unable to communicate to sink at will (in real time)
- Collect data and wait for collector
- Collector not fully trusted



Goal:
authenticate data accumulated during multiple intervals

Two Issues

- Issue 1: Threat of Sensor Compromise
  - Sensor characteristics:
    - Low cost
    - No tamper resistance
  - Adversarial and unattended environment
  - Forward Security:
    - Protect pre-compromise data
    - Periodic key evolution
- Issue 2: Storage and Communication Overhead
  - Sensors have limited on-board storage
  - Forward-security requires one authentication tag per message

Forward-Secure Sequential Aggregate (*FssAgg*) Authentication

- To reconcile minimal storage overhead with mitigating potential secret key compromise
- Allows signer to combine multiple authentication tags generated within different time periods into single constant-size tag
  - Compromise of current key does not endanger authenticity of pre-compromise data
  - Verification of aggregate verifies every component signature

Definition

Component algorithms:

1. FssAgg.Kg: key generation, generate the initial signing key K_0 and the verification key VK

2. FssAgg.Asig: aggregate sign, generate a FssAgg signature $\sigma_{1,i}$ on message m_i and aggregate-so-far FssAgg signature $\sigma_{1,i-1}$ with k_i

3. FssAgg.Upd: key update, generate K_i from K_{i-1}, must be a one way function, and securely erase K_{i-1}

4. FssAgg.Aver: aggregate verify, verify a FssAgg signature with the verification key VK, accept or reject

Properties

1. **Correctness**: any FssAgg signature produced by Asig must be accepted by Aver.

2. **Unforgeability**: without the knowledge of any signing keys, no adversary can compute an FssAgg signature

3. **Forward-security**: No adversary who breaks in i-th time period can generate a valid signature containing a new signed message for any period j<i

MAC-based Scheme

1. FssAgg.Kg: any symmetric key generation algorithm to generate *k*-bit secret *s* and set K_0=VK=*s*

2. FssAgg.Asig: for new message m_i: $a)\ \sigma_i = MAC(K_i, m_i)$

$b)\ \sigma_{1,i} = H(\sigma_{1,i-1} || \sigma_i)$

3. FssAgg.Upd:

a) $K_{i+1} = H(K_i)$

b) Remove K_i, move to time i+1

4. FssAgg.Aver: To verify $\sigma_{1,i}$:

a) Re-compute (from VK): K_1 … K_i,

b) Re-compute $\sigma^c_{1,i}$

c) Compare $\sigma^c_{1,i} \stackrel{?}{=} \sigma_{1,i}$

MAC-based Scheme

- Fast and space-efficient
- But:
  - Either collector cannot authenticate tags

    or
  - Collector can cheat
- Two MAC-based aggregates?
  - one for collector and one for sink
  - or signatures…

Signature-based Scheme

Boneh/Lynn/Shacham. Asiacrypt 2001
Boneh/Gentry/Lynn/Shacham, Eurocrypt 2003

Based on BLS/BGLS signature scheme; works on groups with bilinear map **e:** $G_1 x\ G_2$ --> G_T where : 1) G_1 and G_2 groups of order p; 2) $|G_1|=|G_2|=|G_T|$; 3) g_1, g_2: generator of G_1, G_2

1. FssAgg.Kg: pick random x_0 from Z_p, Compute pairs (x_i, v_i) s.t. $x_i=H(x_{i-1})$, and $v_i=g_1^{x_i}$, set K0=x0, VK={v_i}

2. FssAgg.Asig: given new message m_i, current key K_i, and aggregate-so-far $\sigma_{1,i-1}$

$$a)\ \ \sigma_i = BLS.sign(k_i, m_i)$$

$$b)\ \ \sigma_{1,i} = \sigma_{1,i-1} \bullet \sigma_i$$

3. FssAgg.Update: $K_{i+1} = H(K_i)$, *remove* K_i

4. FssAgg.Aver: To verify $\sigma_{1,i}$

$$e(\sigma_{1,i}, g_2) \stackrel{?}{=} \prod_{t=1}^{i} e(h_t, v_t)$$

Performance Evaluation

- MAC scheme near-optimal
- Signature scheme is not
  - Signer-friendly
    - Constant private key and signature size
    - Efficient signing and key update
  - Not verifier-friendly
    - Public key size - O(*T*)
    - Costly pairing operations in verification

Summary and Future Work

- Motivated Forward-Secure Aggregate authentication
- Two practical schemes:
  - MAC-based scheme near-optimal
  - Signature-based scheme not (yet) verifier-friendly
- Future work: more efficient schemes!

Thanks!

