Secure Aggregation in a Publish-Subscribe System

Kazuhiro Minami, Adam J. Lee, Marianne Winslett, and Nikita Borisov

Workshop on Privacy in the Electronic Society (WPES'08)

Motivation

- Aggregation of data in Publish-Subscribe systems
- Considers "sum" as the aggregation function
- What is a Publish-Subscribe network?

Pub-Sub Network

- Components:
  - Publishers (Pub)
  - Subscribers (Sub)
  - Routers (Infrastructure Components)
- Properties:
  - Asynchronous
  - Decoupling of Pubs and Subs
  - Many to many communication

Pub-Sub Contd.

- Examples:
  - Stock Market
  - News Network
- Pub-Sub Networks with aggregation of data
  - Electric Power Grid
  - Building Management System (BMS)

Objectives

- To ensure confidentiality and integrity of aggregated data in the network
  - Hide individual data sent by publishers
  - Hide aggregate data from unauthorized nodes
  - Detect modification of data values by routing nodes

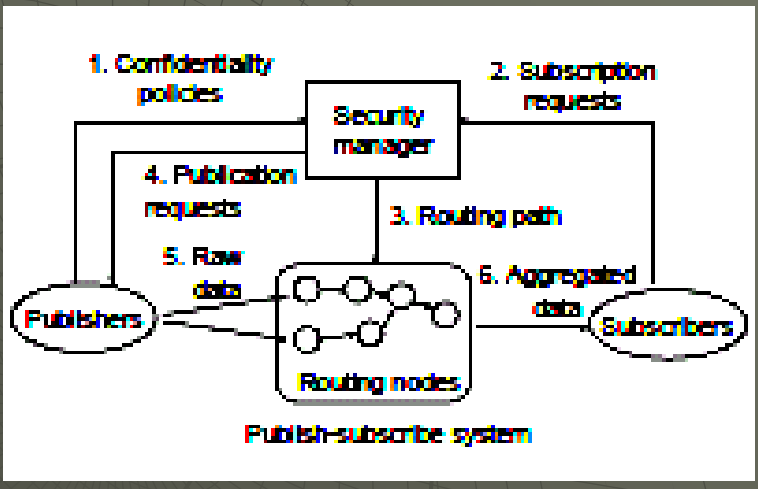
Solution Overview

- Confidentiality
  - Uses PRNG and data splitting
- Integrity
  - Cannot have all the signed raw data
  - Uses properties of Discrete Logarithm
- Let’s look at the system model first !

System Overview

- A set of Principals *P*
- A trusted Security Manager
- Each principal $p_i \in P$ has a key pair (K_i, K_i^{-1})
- Publishers publish data from some value set *Đ*
- Subscription request: A set of pairs in *P X Đ*
  - *E.g.* $\{(p_0,d_0), (p_1,d_1)\}$

System Model



- Each publisher publishes data by synchronizing with other publishers
- Principal P_i's variable d_i at time t is denoted $d_i(t)$
- Subscriber gets $\sum_{i=1..n} d_i(t)$

Confidentiality Policy

- Publisher p_i defines ACL
  - $acl_i(d_i) = \{p_k\}$
  - $acl_i((p_i,d_i),(p_j,d_j))$
- p_k can subscribe for $\{(p_i,d_i),(p_j,d_j)\}$ if
  - $p_k \in acl_i((p_i,d_i),(p_j,d_j))$ and
  - $p_k \in acl_j((p_i,d_i),(p_j,d_j))$

Attack Model

- Publisher's view
  - An adversary colluding with routing nodes to learn unauthorized aggregate
  - A set of routing nodes and subscribers trying to learn individual data values
    - Upto *m-1* routing nodes and *1* subscriber
- Subscriber's view
  - Malicious routing nodes forward incorrect aggregate values
  - Publisher is always trusted

Confidentiality Preserving Aggregation

- Each publisher p_i has data d_i
- Policy:
  - $acl_i(d_i) = \Phi$
  - $acl_i((p_1,d_1),\ldots,(p_n,d_n)) = \{p_{sub}\}$
- Basic concept
  - Publisher generate a random number r_i using a secret q_i and generate
    - $d_i' = d_i - r_i$
  - Split d_i' into m shares and send to *m* routing nodes

- Step 1: Secret Sharing
  - p_i generate a seed q_i randomly
  - Send q_i to subscriber secretly
- Step 2: Publication
  - PRNG: $Z_l \; X \; T \rightarrow Z_p$
    - l is key size, T is set of timestamps
  - $q_i(t_l) = PRNG(q_i, t_l)$
  - $d_i' = d_i - q_i(t_l)$
  - Randomly split d_i' into m shares $d_{i,1}', \ldots, d_{i,m}'$ s.t. $\sum_{j=1..m} d_{i,j}' = d_i'$
  - Send shares to m different routing nodes

- Step 3: Aggregation on Routing Node
  - Receive shares $d_1, \ldots, d_k$ from k publishers or routing nodes
  - Compute $d = \sum_{i=1..k} d_i$
  - Send (d, t_l) to the next routing node
- Step 4: Computation of Sum
  - Router at the root of the aggregation path computes d_{sum}'
  - Subscriber gets d_{sum}'
  - Compute $q_i(t_l) = PRNG(q_i, t_l)$
  - Compute $d_{sum} = d_{sum}' + \sum_{i=1..n} q_i(t_l)$

- Message Complexity
  - Naïve Approach $O(n+r)$
  - Presented Approach $O(nm+r)$
- Guarantees:
  - No coalition of routing nodes can get the sum illegally.
  - If data is split into m parts can protect individual data values (d_i) from $m-1$ routing nodes and the subscriber.

Integrity Preserving Aggregation

- Publisher computes MAC of a value d by as $MAC(d,g) = g^d$, where g is the generator of some multiplicative group G_p
- Every principal knows prime p
- g is a secret among pubs and subs
- Note:
  - $MAC(d_1,g) \; X \; MAC(d_2,g) = MAC(d_1+d_2,g)$

- Step 1: Secret Sharing
  - Security manager generates g and sends to pubs and subs
  - Each publisher generates a seed r_i randomly and send to subscriber
- Step 2: Publication
  - Compute $r_i(t_l) = PRNG(r_i, t_l)$
  - MAC = $c(d_i) = g^{d_i + r_i(t_l)}$
  - Send $d_i, c(d_i)$ to a routing node

- Step 3: Aggregation on Routing Node
  - Receive (*val, MAC*) pairs from pubs or routing nodes
  - Compute the sum
    - $d = \sum_{i=1..k} d_i$
  - Compute the product of MACs
    - $c = \prod_{i=1..k} c_i \ (mod\ p)$
  - Send d, c, t_l to next routing node

- Step 4: Computation and Verification of Sum
  - Receive d_{sum} and c_{sum}
  - Compute $r_i(t_l) = PRNG(r_i, t_l)$ for i = 1..n
  - Accept d_{sum} if $g^{d_{sum}+\sum_{i=1..n} r_i(t_l)}$ matches c_{sum}
- Message Complexity
  - Same as naïve approach
  - Extra 1024-bit MAC required with each message
- Chance of a sub accepting wrong d_{sum} as correct is $(1/p)$

Secure Aggregation

- Combining previous two approaches
- Step 1: Secret Sharing
  - Pubs share q_i, r_i with subs
- Step 2: Publication
  - Compute $q_i(t_l)$ and $r_i(t_l)$
  - Compute $d_i' = d_i - q_i(t_l)$
  - Split d_i' into m shares $d_{i,1}'$, … randomly
  - Compute $d_i'' = d_i + r_i(t_l)$
  - Split d_i'' into m shares $d_{i,1}''$, …

- Compute
  - $c(d_{i,j}'') = (d_{i,j}'', g)$ for $j = 1..m$
- Send the (*value*, *MAC*) pairs $(d_{i,1}', c(d_{i,j}''))$ to m different routing nodes

- Step 3: Aggregation
  - Receive (d_i, c_i) from publishers/routing nodes
  - Compute sum $d = \sum_{i=1..k} d_i$
  - Compute $c = \prod_{i=1..k} c_i \pmod p$
  - Send (d,c) to next routing node

- Step 4: Compute Sum and Verify
  - Compute $r_i(t_l)$ and $q_i(t_l)$ for $i = 1..n$
  - Compute $d_{sum} = d_{sum}' + \sum_{i=1..n} q_i(t_l)$
  - Accept d_{sum} if
    - $c_{sum} = g^{d_{sum} + \sum_{i=1..n} r_i(t_l)}$
- Guarantees:
  - As earlier

- In the previous approach
  - If a subscriber colludes with routing node system is vulnerable
  - Knowing *g* routing nodes can modify data
- Solution
  - In each round security manager generate new g_l
  - Initially only publishers know g_l
  - When all the subscribers receive $d_{sum}(t_l)$, g_l given to subscribers
  - System operates in lock-step

Conclusion

- Consider only "sum" as the aggregate function
- Uses property of Discrete Logarithm to maintain Integrity
- Pub-sub model is very limited
- Does not explain how to build aggregation path (tree)
- Responsibility on security manager