Secure Aggregation in a Publish-Subscribe System

Kazuhiro Minami, Adam J. Lee, Marianne Winslett, and Nikita Borisov
University of Illinois at Urbana-Champaign
UIUCDCS-R-2008-2968
{minami, adamlee, winslett}@cs.uiuc.edu, nikita@uiuc.edu

May 20, 2008

Abstract

A publish-subscribe system is an information dissemination infrastructure that supports many-to-many communications among publishers and subscribers. In many publish-subscribe systems, in-network aggregation of input data is considered to be an important service that reduces the bandwidth requirements of the system significantly. In this paper, we present a scheme for securing the aggregation of inputs to such a publish-subscribe system. Our scheme—which focuses on the additive aggregate function *sum*—preserves the confidentiality and integrity of aggregated data in the presence of untrusted routing nodes. Our scheme allows a group of publishers to publish aggregate data to authorized subscribers without revealing their individual private inputs to either the routing nodes or the subscribers. In addition, our scheme allows subscribers to verify that routing nodes perform the aggregation operation correctly. We use a message authentication code (MAC) scheme based on the discrete logarithm property to allow subscribers to verify the correctness of aggregated data without receiving the digitally-signed raw data used as input to the aggregation. In addition to describing our secure aggregation scheme, we provide formal proofs of its soundness and safety.

1 Introduction

A publish-subscribe (pub-sub) system [2, 3, 18, 21, 17] is an information dissemination infrastructure that supports many-to-many communications between entities in a wide-area network. In a pub-sub system, *publishers* submit information to the system, while *subscribers* can register to receive publications of interest. Data is routed through a network of *routing nodes* that form an overlay network in the system. Pub-sub systems scale to handle large volumes of data from many applications by establishing routing paths that efficiently deliver messages to subscribers while eliminating duplicate messages along those paths.

In this paper, we are particularly interested in pub-sub systems that support *in-network aggregation*, in which routing nodes perform hierarchical aggregation on data that is published to the system. In-network aggregation is useful for applications that monitor the state

of wide-area control systems—such as the electric power grid [22] and building management systems (BMSs) [10]—by collecting individual readings from an array of sensors and other devices. The sensors and other devices monitoring the control system act as publishers that push measurements to the pub-sub system, while monitoring applications act as subscribers that later receive those events from the system. In some cases, sensors deployed over a wide physical area can generate data at very high rates. For example, phasor measurement units [19] used in the electric power grid generate data many times per second. Since most monitoring applications make control decisions based on aggregated data computed from raw sensor data, the demanding bandwidth and latency requirements of the pub-sub system can be reduced through the use of hierarchical in-network aggregation.

Safely supporting hierarchical in-network aggregation in a pub-sub system requires that we address two important security issues. First, publishers should be able to protect their individual inputs from potentially untrusted routers and subscribers. For example, in the power grid, utilities must hide their market sensitive input data from their competitors. Similarly, in BMS systems, the occupancy of a certain room in a building could reveal the occupants' private activities, while aggregate occupancy information for sections of the of the building is likely to be safe to disclose. Since routing nodes in a wide-area pub-sub system are typically managed by entities in different administrative domains, we must assume that publishers do not trust routing nodes in other domains in terms of the confidentiality of their published data. Therefore, our secure aggregation protocol should allow untrusted routing nodes to perform aggregation of raw data without learning the private input values.

Second, subscribers should have some assurance regarding the authenticity and integrity (correctness) of the aggregate data that they receive. Data integrity is critical for monitoring applications used in control systems, as these systems make control decisions based upon aggregated data received from the pub-sub system. Without this property, malicious routers could modify the data aggregated by the system, thereby tricking monitoring applications into making unsafe control decisions. Ensuring this integrity property on aggregated data is challenging because subscribers may not completely trust the routing infrastructure used by system. Furthermore, since thousands of publishers might contribute to a single aggregate data item, subscribers require a means of verifying the correctness of aggregate data without checking signed raw data, as this defeats the purpose of in-network aggregation.

In this paper, we present a secure aggregation protocol for the additive aggregate function *sum*. We believe that this is a reasonable first step towards the general secure aggregation problem for publish-subscribe systems because we can reduce other useful aggregation functions—such as *average*, *count*, *variance*, and *standard deviation*—from the *sum* function. Our protocol allows a collection of publishers to publish an aggregate result derived from their individual private inputs that is released only to authorized subscribers. Thus, such a publish-subscribe system enables users in the system to share useful statistical information without compromising the confidentiality of individual raw data. In addition, we apply a message authentication code (MAC) scheme based on the discrete logarithm property to allow subscribers to verify the correctness of aggregated results. Our scheme eliminates the necessity of providing a subscriber with the signed raw data used as input

to the aggregation, and enables each subscriber to verify the correctness of aggregated data using an aggregated MAC of a constant size. We prove that our protocol satisfies the soundness and safety requirements of both publishers and subscribers.

The rest of the paper is organized as follows. We describe our system and attack models for pub-sub systems in Section 2. We then present our aggregation protocol and proofs of its safety and security properties in Section 3. We discuss a possible way of extending our protocol to reduce our trust assumptions in Section 4. We cover related work in Section 5 and present our conclusions and directions for future work in Section 6.

2 System model

2.1 System overview

We assume that a publish-subscribe system consists of a set of routing nodes as well as a trusted security manager node, while publishers and subscribers exist as applications outside of the pub-sub system. We assume that every publisher, subscriber, and routing node is managed by some principal in the set $\mathcal{P}$ of all principals. The security manager is a central authority that is trusted by publishers and subscribers to coordinate the system. Each principal $p_i \in \mathcal{P}$ maintains a public key pair (K_i, K_i^{-1}), and can obtain the public keys of other principals using a PKI or some other key distribution service. We assume that publishers publish data items from some value set $\mathcal{V}$.

Figure 1 describes the interactions between publishers, subscribers, and the components of the publish-subscribe system. First, each publisher notifies the security manager of the confidentiality policies protecting the aggregation of its data values. As a result, the security manager has a complete view of all data confidentiality policies. When a principal wishes to begin a subscription, he issues a subscription request to the security manager. A subscription request is a set of (principal, variable) pairs in $\mathcal{P} \times \mathcal{V}$ that represents the sum of variables that is to be computed. For example, a subscription request for the sum of publisher principal p_0's variable v_0 and p_1's variable v_1 is represented as $\{(p_0, v_0), (p_1, v_1)\}$.

After a subscription request is issued, the security manager checks whether the subscription request satisfies the confidentiality policies of the publishers owning the variables comprising the subscription request. If the appropriate confidentiality policies are satisfied, the security manager computes a hierarchical routing and aggregation path from the publishers to the subscriber, which is then sent to the appropriate routers. Exactly how the security manager should compute this routing path is out of the scope of this paper—our secure aggregation protocol is independent of the routing path computation algorithm. At this point, the publishers can send their data to routing nodes, which forward the data while performing in-network aggregation. Finally, each subscriber receives the final aggregated data from the root routing node of the routing path.

Each publisher p_i publishes a variable v_i repeatedly, and we denote variable v_i at time t by $v_i(t)$. When a pub-sub system aggregates multiple variables, it only aggregate variables with the same timestamp. Therefore, if each publisher p_i for $i = 1$ to n publishes a variable

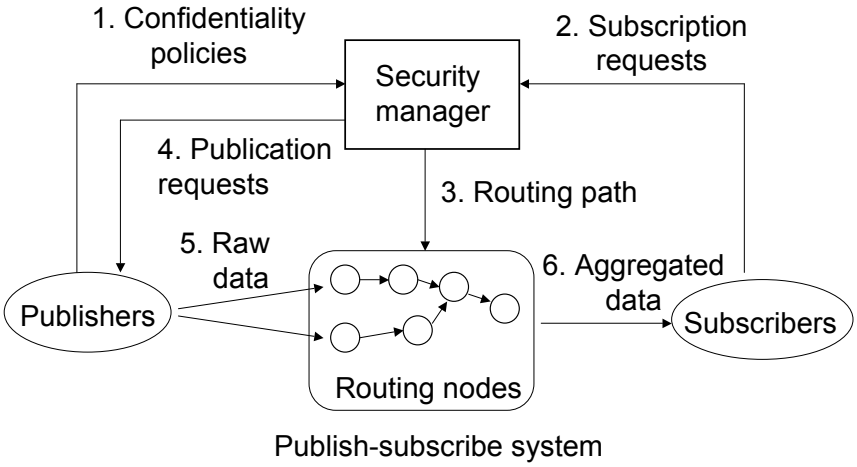

Figure 1: System model. Each arrow is labeled with data transferred between two components.

v_i, a subscriber who subscribes to the sum of those variables will receive $\sum_{i=1}^{n} v_i(t)$ at each time step t.

2.2 Confidentiality policies

Each publisher p_i can define confidentiality policies to limit access to the variables that it maintains. Specifically, a publisher p_i can define an access-control list $acl_i(v_i)$ to limit access to variable v_i. If $acl_i(v_i) = \{p_j\}$, only subscriber p_j can receive p_i's variable v_i. If any other subscriber p_k issues a subscription request for the variable v_i, the security manager maintaining p_i's confidentiality policies rejects p_k's request.

Each publisher can also define an access-control list that protects the sum of multiple variables, some of which are maintained by other publishers. For example, p_i can define policy $acl_i((p_i, v_i), (p_j, v_j))$ to protect the sum of p_i's v_i and p_j's v_j. When a subscriber p_k issues a subscription request $\{(p_i, v_i), (p_j, v_j)\}$ for the sum of the two values, p_k must satisfy both p_i and p_j's confidentiality policies on the sum of those two variables; that is, $p_k \in acl_i((p_i, v_i), (p_j, v_j))$ and $p_k \in acl_j((p_i, v_i), (p_j, v_j))$ must hold.

2.3 Attack models

We now consider attacks on the system from the viewpoints of both publishers and subscribers. From the viewpoint of publishers, there are two kinds of adversaries. One is an adversary of colluding routing nodes that attempts to learn an aggregated sum in an unauthorized way. The other is an adversary who tries to learn some publisher's individual data value in an unauthorized way. The second type of adversary could include unauthorized subscribers as well as untrusted routing nodes. Such colluding parties can freely share

messages that they obtain through the process of in-network aggregation. They can also intercept all the messages among all the parties who participate in the aggregation process. In this paper, when we consider an adversary who tries to learn individual data in an unauthorized way, we assume that the adversary consists of up to m routing nodes and subscribers.

In this paper, we do not address inference attacks carried out by subscribers with multiple subscriptions. We assume that each publisher trusts the security manager to make the proper authorization decisions such that subscribers cannot infer unauthorized data from multiple subscriptions.

From the viewpoint of subscribers, attackers are untrusted routing nodes that incorrectly aggregate data values. Such an adversary can be comprised of any number of colluding routing nodes. On the other hand, we assume that publishers are not part of an adversary against subscribers because publishers can always modify the aggregated data by providing malicious input data while otherwise following the aggregation protocol properly. Therefore, if a subscriber p_{sub} subscribes to an aggregated result that includes some publisher p_i's data, p_{sub} implicitly trusts p_i to provide correct input data. We note that each subscriber must trust other subscribers who subscribe to the same subscription not to construct a bogus result by colluding with some routing nodes. We will describe an extension of our protocol that removes this trust assumption in Section 4.

3 Secure aggregation in a pub-sub system

In this section, we describe our protocol incrementally. In Section 3.1, we first present a protocol that preserves publishers' confidentiality while aggregating data. We then present a protocol that ensures the integrity of aggregated data in Section 3.2. Finally, we present an integrated protocol that ensures both the confidentiality and integrity requirements of publishers and subscribers in Section 3.3.

3.1 Confidentiality-preserving aggregation

We now describe an aggregation protocol that preserves the confidentiality of each publisher's private input. Consider the case in which there are n publishers, $p_1, \ldots, p_n$, and a single subscriber p_{sub} involved in the aggregation protocol. Each publisher p_i maintains a private variable v_i and an associated confidentiality policy $acl_i(v_i) = \emptyset$. That is, p_i is not willing to disclose v_i to any other principal. However, every p_i is willing to disclose the sum $\sum_{i=1}^{n} v_i$ to the subscriber p_{sub} and thus defines the confidentiality policy $acl_i((p_1, v_1), \ldots, (p_n, v_n)) = \{p_{sub}\}$. The n publishers will use the pub-sub system to disseminate the sum of their private variables to the subscriber p_{sub}. Recall from Section 2.3 that we must consider attacks launched by coalitions of up to m colluding nodes, including router nodes and the subscriber p_{sub}.

Our protocol protects each principal p_i's private data v_i from unauthorized routing nodes and subscriber p_{sub} by requiring that each p_i splits their value v_i into m shares,

which are then sent to m different routing nodes. However, this scheme alone is insufficient for protecting the final aggregated sum from unauthorized routers, since eventually a single routing node will compute the final aggregation that will be distributed to p_{sub}. Therefore, we also require that each publisher p_i generates a random number q_i and publishes m shares of the value $v_i - q_i$. In our protocol, the subscriber p_{sub} will know the method that each p_i uses to choose q_i and thus can reconstruct the actual sum from the aggregate value that they eventually receive from the pub-sub system.

Our confidentiality-preserving aggregation protocol consists of the following steps:

Initial secret sharing. Each publisher p_i generates a secret q_i that is used in conjunction with the hash function $H : \mathbb{Z}_p \times \mathcal{T} \rightarrow \mathbb{Z}_p$ where $\mathcal{T}$ is a set of all timestamps, to generate the sequence of q_i values needed during the publication process. The value p is a large prime number.

1. Each publisher generates a seed q_i randomly and sends q_i to the subscriber p_{sub} secretly.

Note that this secret sharing process need only be performed once at the beginning of a subscription request.

Publication of data. Publisher p_i publishes the value v_i at time t_l as follows:

1. Compute $v_i' = v_i - H(q_i, t_l)$. This step is necessary to protect the sum $\sum_{i=1}^{n} v_i$ from the untrusted routing node that computes the sum of all shares.
2. Update q_i as $H(q_i, t_l)$.
3. Randomly split v_i' into m shares $v_{i,1}', \ldots, v_{i,m}'$ such that $v_i' = \sum_{j=1}^{m} v_{i,j}'$.
4. Send shares $v_{i,1}', \ldots, v_{i,m}'$ to m different routing nodes.

Aggregation on routing nodes. Each routing node receives some number of shares from publishers or other routing nodes and sends the sum of these shares to the next routing node along the aggregation path. (We assume that the security manager determines a routing path for each subscription request.) Each routing node performs the following steps:

1. Receive shares $v_1, \ldots, v_k$ from k publishers and/or routing nodes. Note that each of these shares is associated with the same timestamp t_l.
2. Compute the sum of the shares $v = \sum_{i=1}^{k} v_i$.
3. Send (v, t_l) to the next routing node along the aggregation path specified by the security manager.

Computation of the sum. We assume that the security manager establishes an aggregation path such that there is a single routing node that computes the sum v_{sum}' of all

the shares published by all of the publishers. After the subscriber p_{sub} receives the aggregate value v'_{sum} associated with timestamp t_l, it performs the following steps to compute the sum.

1. Compute sum $v_{sum} = v'_{sum} + \sum_{i=1}^{n} H(q_i, t_l) = \sum_{i=1}^{n} v_i$.
2. Update q_i as $H(q_i, t_l)$ for $i = 1$ to n.

Note that a naive aggregation protocol has a communication overhead of $O(n+r)$ where n is the number of publishers and r is the number of routing nodes. Our confidentiality-preserving aggregation protocol, however, has a communication overhead of the order $O(nm + r)$ because each publisher must send m shares to the pub-sub system. We now make the following claims regarding the security properties of this protocol.

Theorem 1 (Confidentiality of aggregate sum)**.** *No coalition of colluding routing nodes can obtain the sum* $\sum_{i=1}^{n} v_i$.

Proof. Note that a routing node can only obtain the sum $v'_{sum} = \sum_{i=1}^{n} v'_i$ where $v'_i = v_i - H(q_i, t_l)$. Since v'_{sum} has no correlation with the actual sum $v_{sum} = \sum_{i=1}^{n} v_i$, it is impossible for a routing node to learn any information about v_{sum} without knowing the random q_i values shared between each p_i and the subscriber p_{sub}. □

Theorem 2 (Confidentiality of an individual data v_i)**.** *Let m be the number of shares generated by each publisher. No colluding adversary of up to size m that includes routing nodes and the subscriber p_{sub} can obtain any principal p_i's private data v_i.*

Proof. Without loss of generality, we discuss the confidentiality of the variable v_i maintained by publisher p_i. The same argument holds for the other publishers' variables. Publisher p_i splits $v'_i = v_i - H(q_i, t_l)$ into m pieces and send each share to a different routing node. Since all of the messages sent between p_i and the routing nodes are encrypted using pairwise shared keys, the only way to restore v'_i is to obtain all the shares of v'_i from the m routing nodes receiving these shares. Restoring v_i from v'_i also requires the value q_i shared between p_i and p_{sub}. Thus, at least $m + 1$ colluding parties are required to obtain the value of v_i. □

3.2 Integrity-preserving aggregation

We next describe an aggregation protocol that ensures only the integrity of the computed aggregate sum. We describe the protocol using the same example aggregation scenario used in Section 3.1. Our integrity-preserving aggregation protocol allows the subscriber p_{sub} to verify the integrity of the sum by leveraging a homomorphic MAC scheme based on the discrete logarithm property. A publisher generates the MAC of a value v by computing $MAC(v, g) = g^v$, where g is a generator for some multiplicative group G_p of prime order p. We assume that every principal knows the large prime number p that parameterizes G_p, and that the generator g is a shared secret among the publishers and the subscriber p_{sub}. As

we will see, the second assumption ensures that a malicious routing node cannot modify an aggregated sum without being detected by the subscriber p_{sub}. This MAC scheme has the homomorphic property $MAC(v_1, g) \times MAC(v_2, g) = MAC(v_1 + v_2, g)$ since $g^{v_1} g^{v_2} = g^{v_1+v_2}$. Our integrity-preserving aggregation protocol consists of the following steps:

Initial secret sharing. All of the publishers and the subscriber p_{sub} share a secret generator g, which is chosen by the security manager. Therefore, the subscriber p_{sub} needs to trust the security manager not to disclose g to the routing nodes. We assume that publishers will not disclose g to the routing nodes since a malicious publisher can always modify the aggregated sum by providing malicious input data. Each publisher p_i and the subscriber p_{sub} also share a secret seed q_i as was the case in the confidentiality-preserving protocol in Section 3.1.

1. The security manager generates a generator g for the group G_p randomly and sends g to each publisher p_i and subscriber p_{sub} secretly. It is assumed that all principals (including routers) know the prime order p of G_p.
2. Each publisher generates a seed q_i randomly and sends q_i to the subscriber p_{sub} secretly.

Publication of data. This protocol does not preserve the confidentiality of each publisher p_i's variable v_i. However, we still require each publisher p_i to subtract a random number $H(q_i, l)$ from v_i because we cannot allow malicious routing nodes to learn a valid (value, MAC) pair (v, c) where $c = MAC(v, g)$. Otherwise, a malicious routing node could construct a bogus (value, MAC) pair (kv, c^k) for any $k \in \mathbb{N}$ by exploiting the homomorphic property of the MAC scheme. Each publisher thus performs the following steps:

1. Compute $v_i' = v_i - H(q_i, t_l)$.
2. Update q_i as $H(q_i, t_l)$.
3. Compute the MAC $c(v_i) = MAC(v_i, g) = g^{v_i}$.
4. Send the value v_i' and MAC $c(v_i)$ to one of the routing nodes.

Notice that the routing node receiving v_i' can learn v_i by colluding with the subscriber p_{sub}, since p_{sub} knows q_i in this protocol. Hence, this protocol preserves *only* the integrity of the aggregation computation.

Aggregation on routing nodes. During the protocol, each routing node receives some number of (value, MAC) pairs from publishers and/or other routing nodes. The router then computes the sum of those values and the product of those MACs, and passes the results to the routing node along the aggregation path. Each routing node performs the following steps:

1. Receive the (value, MAC) pairs $(v_1, c_1), \ldots, (v_k, c_k)$ from other publishers and/or routing nodes in the system. Note that each pair is associated with the same timestamp t_l.
2. Compute the sum of the shares $v = \sum_{i=1}^{k} v_i$.
3. Compute the product of the MACs $c = \prod_{i=1}^{k} c_i$.
4. Send (v, c) and the timestamp t_l to the next routing node along the aggregation path.

Computation and verification of the sum After subscriber p_{sub} receives v'_{sum} and the MAC c_{sum} from the last routing node in the pub-sub system, p_{sub} computes the sum v_{sum} and verifies its correctness using the MAC c_{sum} as follows:

1. Receive v'_{sum} and c_{sum}.
2. Compute the sum $v_{sum} = v'_{sum} + \sum_{i=1}^{n} H(q_i, t_l) = \sum_{i=1}^{n} v_i$.
3. Update q_i as $H(q_i, t_l)$ for $i = 1$ to n.
4. Accept sum v_{sum} if $g^{v_{sum}} = c_{sum}$ holds.

Our integrity-preserving protocol requires the same number of messages as a simple aggregation protocol that does not ensure the integrity of the aggregated data. However, each message of our protocol must also include a MAC of 1024 bits. We now formally prove that our protocol ensures the integrity of aggregated data.

Theorem 3 (Soundness)**.** *The probability that the subscriber p_{sub} accepts an incorrect $v_{sum} \neq \sum_{i=1}^{n} v_i$ is no more than the probability that an adversary can* randomly *generate a valid (value, MAC) pair (v, c) such that $MAC(v, g) = c$.*

Proof. We consider an adversary who controls *all* of the routing nodes in the pub-sub system. There are two ways for such an adversary to generate a valid (value, MAC) pair. The first way is to learn the secret generator g. In this case, the routing node that sends the pair (v'_{sum}, c_{sum}) to p_{sum} can instead transmit $(v'_{sum} + k, g^k \times c_{sum})$ for any $k \in \mathbb{N}$. The subscriber p_{sub} will accept these modified values as correct because if the original pair is valid (i.e., $c_{sum} = g^{v'_{sum}+r}$ where $r = \sum_{i=1}^{n} H(q_i, t_l)$), then $g^{v'_{sum}+k+r} = g^k \times g^{v'_{sum}+r} = g^k \times c_{sum}$ also holds. The second way is to learn the sum v_{sum} that corresponds to a particular MAC c_{sum}. In this case, the adversary can generate other valid pairs (kv, c^k) for any $k \in \mathbb{N}$.

We claim that the adversary obtains no information about the generator g or the aggregated sum v_{sum} from the information gained during the execution of the integrity-preserving aggregation protocol. The colluding nodes receive from each publisher p_i the value $v'_i = v_i - H(q_i, t_l)$ and the MAC $c(v_i)$. First, we show that the colluding nodes learn nothing about the generator g. Specifically, for any $\hat{g} \in G_p$, there exists a value $\hat{v}$ and a seed $\hat{q}$ such that $MAC(\hat{v}, \hat{g}) = MAC(v_i, g)$ and $v_i - H(q_i, t_l) = \hat{v} - H(\hat{q}, t_l)$. Since $MAC(v_i, g) \in G_p$ and $\hat{g}$ is a generator for G_p, there must exist a $\hat{v}$ such that $\hat{g}^{\hat{v}} = g^{v_i}$.

Given this $\hat{v}$, it is then possible to choose a $\hat{q}$ such that $v'_i = \hat{v} - H(\hat{q}, t_l)$. This implies that the disclosure of v'_i and $c(v_i)$ reveals no information about g.

Next, we show that the routing nodes learn nothing about v_{sum} from v'_{sum} and c_{sum}. Given any fake sum $\hat{v}$, there always exists a seed $\hat{q}$ and generator $\hat{g}$ such that $v'_{sum} = \hat{v} - H(\hat{q}, t_l)$ and $MAC(v_{sum}, g) = MAC(\hat{v}, \hat{g})$. To prove the existence of such a $\hat{q}$ and $\hat{g}$, we first note that there exists a $\hat{q}$ such that $v'_{sum} = \hat{v} - H(\hat{q}, t_l)$. At this point, $\hat{g}$ can be chosen such that $\hat{g} = c_{sum}^{1/\hat{v}}$. As a result, $MAC(\hat{v}, \hat{g}) = \hat{g}^{\hat{v}} = MAC(v_i, g)^{\hat{v}/\hat{v}} = MAC(v_i, g)$. Since the colluding nodes gain no information on generator g or the sum v_{sum} from the aggregation process, the probability of generating a valid (value, MAC) pair is no better than that of randomly guessing. □

3.3 Secure aggregation

We now present a secure aggregation protocol that ensures the integrity of aggregated data while also preserving the confidentiality of both individual input values and the aggregated sum. We develop this new protocol by combining the two previous protocols presented in Sections 3.1 and 3.2. In this protocol, each publisher p_i publishes shares of the MAC of variable v_i as well as shares of the value obtained by subtracting some random number from v_i. The protocol consists of the following steps:

Initial secret sharing. This step is exactly same as the initial secret sharing step in the integrity-preserving aggregation protocol presented in Section 3.2.

Publication of data. In this step, each publisher p_i publishes m shares that sum to the value $v_i - H(q_i, t_l)$ and m other shares whose product is $MAC(v_i, g)$. In particular, each publisher performs the following steps:

1. Compute $v'_i = v_i - H(q_i, t_l)$.
2. Update q_i as $H(q_i, t_l)$.
3. Split v'_i into m shares $v'_{i,1}, \ldots, v'_{i,m}$ randomly such that $v'_i = \sum_{j=1}^{m} v'_{i,j}$.
4. Split v_i into m shares $v_{i,1}, \ldots, v_{i,m}$ randomly such that $v_i = \sum_{j=1}^{m} v_{i,j}$.
5. Compute the MAC $c(v_{i,j}) = MAC(v_{i,j}, g) = g^{v_{i,j}}$ for $j = 1$ to m.
6. Send the (value, MAC) pairs $(v'_{i,1}, c(v_{i,1})), \ldots, (v'_{i,m}, c(v_{i,m}))$ to m different routing nodes.

We split the MAC of v_i into multiple shares to prevent a malicious routing node from colluding with a malicious subscriber to attempt to compute the value v_i from the MAC g^{v_i}. This would be possible if, for example, the expected domain of v_i was small (e.g., if all v_i are expected to be close to some average value).

Aggregation on routing nodes. Each routing node receives shares of values and MACs from publishers and/or other routing nodes. The router then computes the sum of the

value shares and the product of the MAC shares. Each routing node performs the following:

1. Receive the (value, MAC) pairs $(v_1, c_1), \ldots, (v_k, c_k)$, from some set of publishers and/or other routing nodes. Note that each (value, MAC) pair is associated with the same timestamp t_l,
2. Compute the sum of the shares $v = \sum_{i=1}^{k} v_i$.
3. Compute the product of the MACs $c = \prod_{i=1}^{k} c_i$.
4. Send (v, c) and the timestamp t_l to the next routing node along the aggregation path specified by the security manager.

Computation and verification of the sum After the subscriber p_{sub} receives the sum v'_{sum} and the MAC c_{sum} from the last routing node in the aggregation path, they perform the following steps to compute the sum of $v_1, \ldots, v_n$ and verify its correctness:

1. Compute the sum $v_{sum} = v'_{sum} + \sum_{i=1}^{n} H(q_i, t_l) = \sum_{i=1}^{n} v_i$.
2. Update q_i as $H(q_i, t_l)$ for $i = 1$ to n.
3. Accept the sum v_{sum} if $g^{v_{sum}} = c_{sum}$ holds.

This secure aggregation protocol has the same communication complexity as the protocol presented in Section 3.1, but each message also includes a MAC value of 1024 bits.

We now consider the security properties of this secure aggregation protocol. The secure aggregation protocol preserves the confidentiality of the aggregate sum and individual data items as in Theorems 1 and 2. The proofs for this protocol are the same as those for the confidentiality-preserving protocol in Section 3.1, and thus we omit them here. We now prove that the secure aggregation protocol satisfies the soundness property stated in Theorem 3.

Theorem 4 (Soundness)**.** *The secure aggregation protocol satisfies the soundness property stated in Theorem 3.*

Proof. See Appendix A. □

4 Secure aggregation with an authenticated MAC

In the secure aggregation protocol described in Section 3.3, the generator g used in the MAC scheme must be kept secret from the untrusted routing nodes. Otherwise, a malicious router could modify a (v'_{sum}, c_{sum}) pair as was described in the proof of Theorem 3, thereby forcing a subscriber to accept an incorrect aggregate value. Therefore, in that section, we made the assumption that each subscriber trusts the other subscribers not to collude with the untrusted routing nodes. In this section, we show how this assumption can be removed

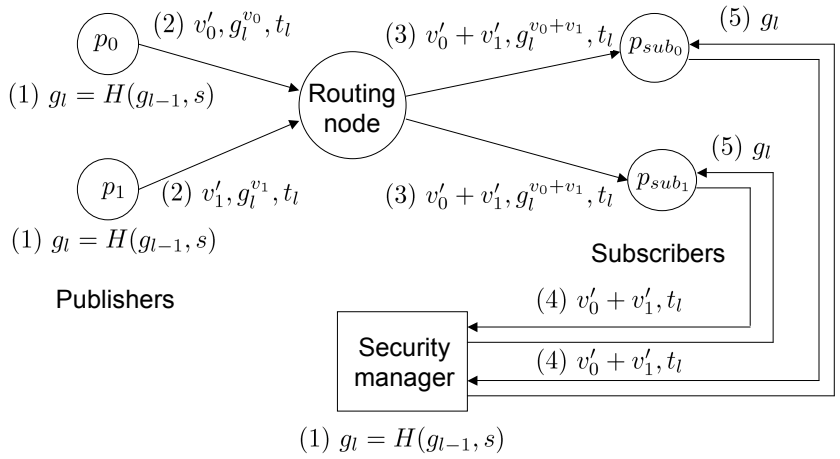

Figure 2: Example of authenticating an aggregated MAC. The arrows shows the sequence of messages. For brevity, we only show the root routing node of a routing path in a pub-sub system.

by taking a *delayed verification* approach to ensuring the integrity of received aggregate values.

Figure 2 illustrates the mechanism used by our approach to providing subscribers with an authenticated MAC. For brevity, we only show a single routing node that sends the final result to the subscribers in Figure 2. In our scheme, the publishers and the security manager share a secret value s that is used to generate a psuedorandom sequence of generators for G_p that will be used by the MAC scheme. At each time t_l, a new generator is created by the publishers and used to commit to values published during the current round of the aggregation. Value and MAC shares are then published and aggregated by the pub-sub system as described in Section 3.3, and eventually received by the subscribers in the system. After a subscriber receives a (v'_{sum}, c_{sum}) pair at time t_l, he notifies the security manager that the value has been received. After the security manager receives such a notification from *every* publisher, it uses the generator g_{l-1} from the previous round and the secret s to create the generator g_l. The generator g_l is then released to all of the publishers, who can then verify the integrity of the sum that was computed.

Notice that the subscribers cannot create the generator g_l themselves using g_{l-1}, as doing so requires knowledge of the secret s, which is shared only between the publishers and the security manager. As a result, *no* subscriber can verify the integrity of the aggregation that they receive during a particular round of the negotiation until *all* subscribers have received the aggregated value computed during that round. By forcing the system to proceed in a lockstep manner, the threat of malicious subscribers colluding with the routing nodes in the system to force other subscribers to accept incorrect aggregate values is clearly eliminated. This benefit comes at the cost of delaying the verification of the sums computed during the protocol; whether this delay is acceptable is application dependent, and may

vary depending upon the environment in which our algorithm is deployed.

5 Related work

In this section, we discuss related research efforts. We first examine existing security solutions for pub-sub systems and next discuss secure aggregation protocols for sensor networks. We finally cover research on verification of aggregated queries on outsourced databases briefly.

5.1 Security in a pub-sub system

Many researchers have explored the security issues that arise in pub-sub systems. Wang et al. [24] describe various security issues with varying trust assumptions in a pub-sub system. In this paper, we focus on the issue of publication confidentiality and integrity under the presence of untrusted routing nodes, to which Wang et al. do not provide any concrete solutions.

Several researchers [25] have studied policy languages and enforcement mechanisms for limiting access to events in pub-sub systems with trusted routing nodes. Miklos [12] provides a policy language that defines access-control policies as filters in a pub-sub system. Zhao and Sturman [25] implement an access-control mechanism as a message filter on trusted routing nodes. Pesonen et al. [14] provides a scheme for delegation-based access control in a pub-sub system involving multiple security domains. Opyrchal et al. [13] develop an efficient key distribution scheme based on techniques of key caching to ensure confidentiality from end-point routing nodes to groups of subscribers. In this paper, we consider policy enforcement in systems with untrusted routing nodes.

Recently, researchers have begun to investigate security issues in pub-sub systems with untrusted routing nodes. However, none of this research addresses security issues associated with aggregated data. Khurana's scheme [11] ensures publication confidentiality against routing nodes by encrypting confidential fields of each event with a key shared between a publisher and subscribers. Pesonen et al. [15] also use encryption on event attributes to protect confidential data from untrusted routing nodes. Raiciu et al. [16] developed several security protocols that allow routers to perform content-based filtering based on equality, keywords, and numeric values while keeping publications and subscriptions confidential from those routers. EventGuard [20] provides a comprehensive solution to various security problems in pub-sub systems that involve untrusted routing nodes. EventGuard ensures publication confidentiality by encrypting events with a shared key shared by publishers and subscribers. EventGuard has a central authority to issue a group key per each topic. EventGuard also ensures publication integrity by requiring publishers to sign their events. However, EventGuard does not address publication confidentiality and integrity for aggregated data.

Ahmad et al. [1] developed a secure additive aggregation protocol in a large-scale overlay network. Their protocol uses an additively homomorphic public-key cryptosystem to

protect confidential data from intermediate aggregation nodes. However, their scheme does not address the issue of integrity discussed in this paper. That is, there is no way for a subscriber to verify that no malicious intermediate node added an encrypted share multiple times to change the value of the sum.

5.2 Secure aggregation in a sensor network

Secure aggregation that ensures the confidentiality of raw data and the integrity of aggregated data has been mainly studied in the context of wireless sensor networks. In a sensor network, there is a single sink node that obtains aggregated data derived from sensor readings. Since the sink node is trusted to read all sensor readings, we do not consider the attack by colluding sensor nodes and the sink node, as we did for colluding parties of routing nodes and a subscriber.

Wagner [23] studies which aggregation functions can be securely computed in the presence of a few compromised nodes providing malicious input data. His model assumes that aggregation is performed in a single sink node of a sensor network. In this paper, we consider the case in which each subscriber trusts the publishers of aggregated data to provide correct inputs, and thus our security model excludes his threat model.

A number of aggregation protocols have been developed to ensure the confidentiality of sensor data from intermediate aggregator nodes. PDA [8] supports additive aggregation using a technique of secret splitting that is similar to ours, while protecting each sensor reading from other sensor nodes. However, PDA does not ensure the integrity of aggregated data. CDA [6] supports additive aggregation using an additively homomorphic encryption scheme to protect confidential sensor data from intermediate nodes that perform aggregation. In CDA, sensor nodes performing aggregation cannot report sensing data. Castelluccia et al. [4] also allows aggregation of encrypted data using an additively homomorphic stream cipher where each sensor node can have a different shared key with a sink node. Therefore, sensor nodes publishing data and aggregation nodes do not have to be disjoint. Solutions based on homomorphic encryption are not applicable to our problem since a malicious routing node that colludes with an unauthorized subscriber can forward an encrypted individual data to the subscriber who can decrypt it.

There are a few aggregation protocols that ensure the integrity of aggregated data. Hu and Evans [9] develop a secure aggregation protocol that ensures the integrity of aggregated data. In their protocol, each sensor node publishes data along with a message authentication code (MAC) constructed using a key generated every time it sends new data. Each node's parent forwards that data and its MAC along with the MAC for the aggregated data it receives to its parent. At every round of the protocol, a base station needs to broadcast the keys used by sensor nodes at the previous round so that each sensor node can verify the MACs that it received from other nodes. If a pair of child-parent nodes in the sensor network is compromised, their protocol cannot ensure the integrity of aggregated data. On the other hand, our scheme ensures integrity under the presence of an arbitrary number of untrusted routing nodes. Chan et al. [5]'s aggregation protocol ensures the integrity of additive aggregation by enabling sensor nodes to construct a commitment tree in a distributed fash-

ion while they perform in-network aggregation. The base station receiving the sum verifies its correctness by sending the root vertex of the commitment tree to all sensor nodes using an authenticated broadcast. If each sensor node can verify that its contribution was added to the sum by checking the commitment tree, the base station accepts the sum. Although Chan's protocol does not involve expensive cryptographic operations, it is not applicable to a pub-sub system where routing nodes are deployed across a wide-area network because the authenticated broadcast would be inefficient in this case.

5.3 Verification of aggregated queries

Haber et al. [7] address the problem of verifying the integrity of aggregate queries on outsourced databases. They develop a verification protocol that allows a user to verify the integrity of the sum of multiple values in a database without seeing those individual values. This query is processed by an untrusted service provider that is different from the trusted database owner. Since a single database owner provides all of the individual values for the sum, their protocol can ensure the integrity of the sum by providing the user with a Merkle hash tree of commitments to the individual values, so that the user can verify the authenticity of those commitments with a digital signature on the root node of the hash tree created by the database owner. This solution of constructing a single digital signature on multiple commitments is not applicable to our problem, since each individual data is provided by a different publisher.

6 Conclusion

In this paper, we presented a secure aggregation protocol for computing the additive function *sum* in a publish-subscribe system. Our protocol allows an aggregated value to be computed from the raw inputs of some number of publishers in a privacy-preserving manner. Secret splitting is used to ensure that as long as no more than m parties collude, no principal's private data value will be leaked. Our protocol further guarantees that the computed aggregate is disclosed only to authorized subscribers and cannot be inferred by the untrusted routing infrastructure that comprises the pub-sub system. In addition, our scheme allows subscribers to verify the correctness of the aggregate value computed by the system by leveraging a homomorphic message authentication (MAC) scheme based on the discrete logarithm property. This MAC scheme allows the correctness of an aggregation to be verified by subscribers and imposes a small constant-size data transmission overhead, regardless of the number of publishers contributing to the aggregate value computed by the system.

In the future, we hope to develop support for other useful aggregate functions, such as *min* and *max*. We also plan to incorporate a fault tolerance mechanism that allows partial aggregates to be computed in the presence of failed publisher nodes. Supporting such a mechanism will require an access control policy language that specifies both disclosure constraints, as well as constraints on the number of failed nodes that will be tolerated by a

given publisher.

References

[1] Waseem Ahmad and Ashfaq Khokhar. Secure aggregation in large scale overlay networks. *Global Telecommunications Conference (GLOBECOM '06)*, pages 1–5, November 2006.

[2] David E. Bakken, Carl H. Hauser, Harald Gjermundrod, and Anjan Bose. Towards more flexible and robust data delivery for monitoring and control of the electric power grid. Technical Report TR-GS-009, Washington State University, May 2007.

[3] Antonio Carzaniga, David S. Rosenblum, and Alexander L. Wolf. Design and evaluation of a wide-area event notification service. *ACM Transactions on Computer Systems*, 19(3):332–383, August 2001.

[4] Claude Castelluccia, Einar Mykletun, and Gene Tsudik. Efficient aggregation of encrypted data in wireless sensor networks. In *The Second Annual Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous 2005)*, pages 109–117, July 2005.

[5] Haowen Chan, Adrian Perrig, and Dawn Song. Secure hierarchical in-network aggregation in sensor networks. In *Proceedings of the 13th ACM conference on Computer and communications security (CCS 2006)*, pages 278–287, New York, NY, USA, 2006. ACM.

[6] Joao Girao, Markus Schneider, and Dirk Westhoff. On concealed data aggregation in wireless sensor networks. In *Proceedings of IEEE International Conference on Communication (ICC 2005)*, May 2005.

[7] Stuart Haber, William Horne, Tomas Sander, and Danfeng Yao. Privacy-preserving verification of aggregate queries on outsourced databases. Technical Report HPL-2006-128, HP Labs, December 2006.

[8] Wenbo He, Lue Liu, Hoang Nguyen, Klara Nahrstedt, and Tarek Abdelzaher. Pda: Privacy-preserving data aggregation in wireless sensor networks. *26th IEEE International Conference on Computer Communications (INFOCOM 2007)*, pages 2045–2053, May 2007.

[9] Lingxuan Hu and David Evans. Secure aggregation for wireless networks. In *SAINT-W '03: Proceedings of the 2003 Symposium on Applications and the Internet Workshops (SAINT'03 Workshops)*, page 384, Washington, DC, USA, 2003. IEEE Computer Society.

[10] Wolfgang Kastner, Georg Neugschwandtner, Stefan Soucek, and Michael H. Newmann. Communication systems for building automation and control. *Proceedings of the IEEE*, 93(6):1178–1203, June 2005.

[11] Himanshu Khurana. Scalable security and accounting services for content-based publish/subscribe systems. In *Proceedings of the 2005 ACM symposium on Applied computing*, pages 801–807, New York, NY, USA, 2005. ACM Press.

[12] Zoltan Miklos. Towards an access control mechanism for wide-area publish/subscribe systems. In *Proceedings of the 22nd International Conference on Distributed Computing Systems*, pages 516–524, Washington, DC, USA, 2002. IEEE Computer Society.

[13] Lukasz Opyrchal and Atul Prakash. Secure distribution of events in content-based publish subscribe systems. In *Proceedings of the 10th conference on USENIX Security Symposium*, pages 21–21, Berkeley, CA, USA, 2001. USENIX Association.

[14] Lauri I. W. Pesonen, David M. Eyers, and Jean Bacon. A capability-based access control architecture for multi-domain publish/subscribe systems. In *Proceedings of the International Symposium on Applications on Internet (SAINT '06)*, pages 222–228, Washington, DC, USA, 2006. IEEE Computer Society.

[15] Lauri I. W. Pesonen, David M. Eyers, and Jean Bacon. Encryption-enforced access control in dynamic multi-domain publish/subscribe networks. In *Proceedings of the 2007 inaugural international conference on Distributed event-based systems*, pages 104–115, New York, NY, USA, 2007. ACM.

[16] Costin Raiciu and David S. Rosenblum. Enabling confidentiality in content-based publish/subscribe infrastructures. *Securecomm and Workshops*, pages 1–11, 2006.

[17] Venugopalan Ramasubramanian, Ryan Peterson, and Emin Gun Sirer. Corona: A high performance publish-subscribe system for the world wide web. In *Proceedings of the 3rd Symposium on Networked Systems Design and Implementation (NSDI)*, May 2006.

[18] Robbert Van Renesse, Kenneth P. Birman, and Werner Vogels. Astrolabe: A robust and scalable technology for distributed system monitoring, management, and data mining. *ACM Transactions on Computer Systems*, 21(2):164–206, 2003.

[19] Jr. Robert O. Burnett, Marc M. Butts, and Patrick S. Sterlina. Power system applications for phasor measurement units. *Computer Applications in Power, IEEE*, 7(1):8–13, 1994.

[20] Mudhakar Srivatsa and Ling Liu. Secure event dissemination in publish-subscribe networks. In *Proceedings of the 27th International Conference on Distributed Computing Systems (ICDCS)*, page 22, Washington, DC, USA, 2007. IEEE Computer Society.

[21] Robert Strom, Guruduth Banavar, Tushar Chandra, Marc Kaplan, Kevan Miller, Bodhi Mukherjee, Daniel Sturman, and Michael Ward. Gryphon: An information flow based approach to message brokering. In *International Symposium on Software Reliability Engineering (ISSRE '98)*, November 1998.

[22] Kevin Tomsovic, David E. Bakken, Vaithianathan Venkatasubramanian, and Anjan Bose. Designing the next generation of real-time control, communication, and computations for large power systems. *Proceedings OF THE IEEE*, 93(5):965–979, 2005.

[23] David Wagner. Resilient aggregation in sensor networks. In *Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*, pages 78–87, New York, NY, USA, 2004. ACM.

[24] Chenxi Wang, Antonio Carzaniga, David Evans, and Alexander L. Wolf. Security issues and requirements for Internet-scale publish-subscribe systems. In *Proceedings of the 35th Annual Hawaii International Conference on System Sciences*, Big Island, Hawaii, January 2002.

[25] Yuanyuan Zhao and Daniel C. Sturman. Dynamic access control in a content-based publish/subscribe system with delivery guarantees. In *Proceedings of the 26th IEEE International Conference on Distributed Computing Systems (ICDCS)*, page 60, Washington, DC, USA, 2006. IEEE Computer Society.

A Proof of Theorem 4

Proof. To prove this claim, we consider an adversary who controls all of the routing nodes in the pub-sub system. Using an argument similar to that used in Theorem 3, we can show that such an adversary obtains no information about the generator g or the aggregated sum v_{sum} during the execution of the protocol.

The colluding nodes receive from each publisher p_i the shares $(v'_{i,1}, \ldots, v'_{i,m})$ of the value v_i and the shares $(MAC(v_{i,1}, g), \ldots, MAC(v_{i,m}, g))$ of the MAC $MAC(v_i, g)$ to value v_i. The routing nodes learn nothing about the generator g from these shares, since for every generator $\hat{g}$, there exists some secret seed $\hat{q}_i$ and some value $\hat{v}_i$ such that $v'_i = \hat{v}_i - H(\hat{q}, t_l)$ and $MAC(v_{i,k}, g) = MAC(\hat{v}_{i,k}, \hat{g})$ for $k = 1$ to m. To prove the existence of such a $\hat{v}_i$ and $\hat{q}_i$, we first note that by the definition of a generator, there exists some $\hat{v}_{i,1}, \ldots, \hat{v}_{i,m}$ such that $MAC(v_{i,k}, g) = MAC(\hat{v}_{i,k}, \hat{g})$ for $k = 1$ to m. Given these m values, $\hat{v}_i$ can be chosen as $\sum_{k=1}^{m} \hat{v}_{i,k}$. At this point, it is possible to choose $\hat{q}_i$ such that $v'_i = \hat{v}_i - H(\hat{q}_i, t_l)$.

Similarly, the routing nodes learn nothing about the value v_i, since given any value $\hat{v}_i$, the existence of another seed $\hat{q}_i$ and generator $\hat{g}$ that can be used to generate the same sets of shares is always guaranteed. To prove the existence of such a $\hat{q}_i$ and $\hat{g}$, we first note that the existence of a $\hat{q}$ such that $v'_i = \hat{v}_i - H(\hat{q}, t_l)$ is guaranteed. By an argument similar to that used in the proof of Theorem 3, there exists a $\hat{g}$ such that $\hat{g}^{\hat{v}_i} = \prod_{k=1}^{m} MAC(v_{i,k}, g) = g^{v_i}$. Given this $\hat{g}$, there exist values $\hat{v}_{i,1}, \ldots, \hat{v}_{i,m-1}$ such that $MAC(v_{i,k}, g) = MAC(\hat{v}_{i,k}, \hat{g})$ for $k = 1$ to $m - 1$ by the definition of a generator. The value $\hat{v}_{i,m}$ can then be chosen as $\hat{v}_i - \sum_{k=1}^{m-1} \hat{v}_{i,k}$. We can show that $MAC(\hat{v}_{i,m}, \hat{g}) = MAC(v_{i,m}, g)$ as follows:

$$MAC(\hat{v}_{i,m}, \hat{g}) \quad = \quad \hat{g}^{\hat{v}_{i,m}}$$

$$
\begin{aligned}
&= \hat{g}^{\hat{v}_i - \sum_{k=1}^{m-1} \hat{v}_{i,k}} \\
&= \hat{g}^{\hat{v}_i} / \hat{g}^{\sum_{k=1}^{m-1} \hat{v}_{i,k}} \\
&= \prod_{k=1}^{m} MAC(v_{i,k}, g) / \prod_{k=1}^{m-1} MAC(v_{i,k}, g) \\
&= MAC(v_{i,m}, g)
\end{aligned}
$$

Since the colluding nodes gain no information on generator g, the individual values v_i, or the sum v_{sum} from the aggregation process, the probability of generating a valid (value, MAC) pair is no better than that of randomly guessing such a pair. □