PRIEST: A Private Subscription Scheme in Publish-Subscribe Systems [*]

Weifeng Chen and Don Towsley

Department of Computer Science
University of Massachusetts, Amherst
{*chenwf, towsley*}@*cs.umass.edu*

Technical Report 2004-81

Abstract

In a publish/subscribe (pub/sub) system, messages are sent to the network and filtered by the network according to subscribers' interests. When the pub/sub infrastructure is untrusted, it is desirable to keep both the sensitive messages and the interests secret from the pub/sub network. In this paper, we formulate this goal as the *private subscription problem*, which is then shown to be at least as hard as the single-database private information retrieval problem introduced in [7]. We then describe cryptographic schemes to keep both published messages and subscribers' interests secret from the network. Our schemes are computationally secure and support both channel-based filtering and content-based filtering, the latter supporting both primitive-event detections and composite-event detections. The algorithms we present are efficient in that they are based on symmetric encryptions requiring $O(n)$ cipher operations for a message of length n.

Key words: Private subscription, Security

[*]This research has been supported in part by the NSF under grant awards UF-EIES-0205003-UMA and EIA-0080119. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

1 Introduction

In a publish/subscribe (pub/sub) system, *publishers*, the entities providing events, advertise information about events and subsequently publish events to the network. *Subscribers*, entities interested in receiving events, subscribe interests and, consequently, receive matched events. The network, typically composed of *brokers*, provides the functionalities of advertisement/subscription processing, and event storing, matching and delivering.

Wang *et. al.* discussed in [24] several security issues and requirements for Internet-scale pub/sub systems. *Information confidentiality* enables a publisher to keep sensitive events secret from a not necessarily trusted pub/sub network. Information confidentiality is attractive in "an application that allows users (publishers) to post their resumes and sells resume information to interested human resource offices (subscribers) [24]." *Subscription confidentiality* allows a subscriber to obtain interested data without revealing its interests, which may disclose sensitive information about the subscriber, to the publishers and the network. The authors consequently recommended the combination of information and subscription confidentiality against the network to achieve a strong level of user privacy, although they did not provide a solution for this problem.

Such confidentiality assurance is desirable in the Transnational Digital Government (TDG) project [10]. The goal of the TDG project is to construct a pub/sub system based on active databases [17] to facilitate information sharing among the Organization of American States (OAS), Belize and the Dominican Republic. Consider the following scenario in the TDG project. Subscribers in Belize are interested in being notified about particular events generated by publishers from the Dominican Republic, via brokers controlled by the OAS. Additionally, the publishers want to keep events secret from the brokers, and the subscribers wish to keep their interests secret from both the brokers and the publishers. One approach for providing information security is the following. A publisher from the Dominican Republic can encrypt all generated events using a key known only to the subscribers and sends all encrypted events to the network. Evidently, this approach keeps all events secret from the brokers. However, it is not efficient at all since a subscriber has to receive all events, no matter whether an event matches its interests or not. Alternatively, subscribers' interests may be sent to the publishers and, consequently, only matched events are encrypted and sent by the publishers. The second approach is quite efficient, but violates the subscription confidentiality since the publishers will know the interests of the subscribers.

In this paper, we refer to the problem of ensuring information confidentiality and subscription confidentiality as the *private subscription problem*. We then propose a PRIvatE SubscripTion (PRIEST) scheme that solves this problem. More specifically, publishers and subscribers share a set of secret keys and all events generated by the publishers are encrypted by these keys and sent to the network. Subscribers also encrypt their interests and send their interests to the network. An intermediate node in the network that does not know any key, is still able to detect events matching the interests. Our scheme supports both channel-based systems and content-based systems, the latter supporting both primitive-event detections and composite-event detections. The proposed scheme is efficient in the sense that it is based on symmetric encryptions and a subscriber only receives matched events. For a message of length n, the scheme requires $O(n)$ cipher operations and hence is practical to use.

The rest of the paper is organized as follows. In Section 2 we provide some necessary definitions, describe the system model, and formulate the private subscription problem. Section 3 presents our scheme to solve the private subscription problem. Section 4 discusses several additional issues. We briefly review the related work in Section 5 and finally conclude in Section 6.

2 Preliminaries

In this section, we first introduce the notations and the computationally-secure assumptions that are used in this paper. We then describe the considered pub/sub model, followed by the formulation of the private subscription problem.

2.1 Notations

Throughout this paper, we will use the following notation unless otherwise stated. Letters in lower case are used to represent plaintext and keys, e.g., $x = \{0,1\}^{n_x}$ and $k = \{0,1\}^{n_k}$. Corresponding ciphertext is denoted by letters in upper case. Encryption, decryption and hash functions are represented by letters in boldface (**S, P**, etc.). Letters in calligraphic case (e.g., $\mathcal{C}$) are used to represent sets. We write $\langle x, y \rangle$ for the concatenation of x and y, and $x \oplus y$ for the bitwise XOR of x and y[1].

[1]For convenience, if the binary representations of x and y have different lengths, a certain number of 0s are padded at the most significant bits of the smaller one, e.g., $101 \oplus 10110 = 00101 \oplus 10110$.

2.2 Intractability assumptions

2.2.1 One-way collision-resistant hash functions

Let $\mathbf{H} : \{0,1\}^* \mapsto \{0,1\}^{n_h}$ be a one-way collision-resistant hash function that maps an input of arbitrary length to an output of length n_h. We assume that $\mathbf{H}$ has the following properties [1], based on which we will construct a *computationally-secure* PRIEST scheme:

- *preimage resistance* — for an output $y = \mathbf{H}(x)$, it is computationally infeasible to find any preimage x' such that $\mathbf{H}(x') = y$. Formally, for every constant c, and every family of polynomial-size circuits $\mathbf{C}_m(\cdot)$, there exists an integer m_0 such that for all $m > m_0$, Pr$[\mathbf{C}_m(y) = x] < 1/2 + 1/m^c$;
- *collision resistance* — it is computationally infeasible to find any two distinct inputs x and x' such that $\mathbf{H}(x) = \mathbf{H}(x')$.

2.2.2 Factorization assumptions

The *integer factorization problem* (FACTORING) is the following [1]: "given a positive integer N, find its prime factorization; that is, write $N = p_1^{e_1} p_2^{e_2} \ldots p_k^{e_k}$ where the p_i are pairwise distinct primes and each $e_i \geq 1$." It is known that the FACTORING problem is an intractable problem.

2.3 Models

After advertising information about events, a publisher publishes events to the network. If a subscriber is interested in receiving particular events, it sends its interests to the network. Different proposed schemes in the literature [4, 11, 23] can be applied to set up routes for event distribution from a publisher to subscribers. For simplicity, we model the network as a black box that detects events matching subscribers' interests and delivers matched events. Without loss of generality, we only consider a single publisher connecting a single subscriber via the modelled network shown in Figure 1.

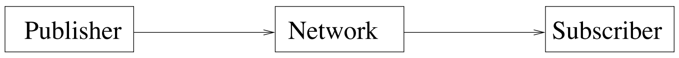


Figure 1: A simple model for pub/sub systems.

We can classify pub/sub systems into *channel*-based (or subject-based) systems and *content*-based systems [4].

2.3.1 Channel-based systems

In a channel-based system, publishers publish messages, each of which belongs to one of a fixed set of channels (alternatively referred to as groups or topics); subscribers subscribe to messages by targeting a channel and, consequently, receive all messages that are associated with that channel. Let $\mathcal{D}$ be the set of IDs of predefined channels. A message m is modelled as a pair $m = (d, x) \in \mathcal{D} \times \{0,1\}^*$, where $d \in \mathcal{D}$ and x is the content of this message. In this case, a subscriber's interest is specified as a set of channel IDs, denoted as $\mathcal{I} \subseteq \mathcal{D}$.

Definition 2.1. *Given a publisher, a network and a subscriber shown in Figure 1, a PRIvatE SubscripTion (PRIEST) scheme in a channel-based system consists of*

(1) A service, $\mathbf{S} : \mathcal{I} \mapsto \{0,1\}^*$*, for the subscriber to encrypt each ID* $d \in \mathcal{I}$ *in its interest as* $\mathbf{S}(d) = D_d$*;*

(2) A service, $\mathbf{P} : \mathcal{D} \times \{0,1\}^* \mapsto \{0,1\}^*$*, for the publisher to encrypt a message* m *as* $\mathbf{P}(m) = M$*;*

(3) A service, $\mathbf{R} : \{0,1\}^* \mapsto \mathcal{D} \times \{0,1\}^*$*, for the subscriber to recover the ciphertext, i.e.,* $\mathbf{R}(\mathbf{P}(m)) = m$.

These services should satisfy

Correctness: *For any message* $m = (d, x)$*,* m *is received by the subscriber if and only if* $d \in \mathcal{I}$*. Namely, a subscriber receives all messages associated with the channels that the subscriber's interest targets.*

Privacy: *(1) Given an encrypted channel ID* $D = \mathbf{S}(d)$ *in the subscriber's interest, it is computationally infeasible for the network to infer* d*. More formally, for all constants* c *and for all polynomial-size families of circuits* $\mathbf{C}_m(\cdot)$*, there exists an integer* m_0 *such that for all* $m > m_0$*,* $Pr[\mathbf{C}_m(D) = d] < 1/2 + 1/m^c$*; (2) Given an encrypted message* $M = \mathbf{P}(m)$ *where* $m = (d, x)$*, it is computationally infeasible for the network to infer* d *or* x*.*

Since channel-based systems lack scalability and expressiveness [4] (e.g., limited number of predefined channels), recent research in pub/sub systems focuses on content-based systems [4, 23].

2.3.2 Content-based systems

Intuitively, PRIEST schemes for channel-based systems are simpler than schemes for content-based systems. Both the pub/sub architectures SIENA [4] and Gryphon [23] support content-based subscriptions. In this paper, an *event notification* (or simply an *event*) is modelled as a set of typed attributes, following the notations in SIENA. Each individual attribute has a *type*, a *name* and a *value*. An attribute type is chosen from “a predefined set $\mathcal{T}$ of primitive types commonly found in programming languages and database query languages [4],” e.g., string, float, integer. We denote an attribute as $\alpha = (type, name, value)$ and an event as $e = \{\alpha_1, \ldots, \alpha_n\}$. A subscriber’s interest is specified as an *event filter* (or simply a *filter*) consisting of a set of constraints. A constraint is a tuple consisting of a type, a name, a binary predicate *operator* chosen from a predefined set $\mathcal{O}$, and a value for an attribute. An attribute $\alpha = (type_\alpha, name_\alpha, value_\alpha)$ matches an attribute constraint $\phi = (type_\phi, name_\phi, operator_\phi, value_\phi)$ if and only if $(type_\alpha = type_\phi) \wedge (name_\alpha = name_\phi) \wedge operator_\phi(value_\alpha, value_\phi)$, which is denoted as $\alpha \prec \phi$. An event e *matches* a constraint ϕ, denoted as $e \prec_S^N \phi$ in [4], if and only if $\exists \alpha_i \in e : \alpha_i \prec \phi$.

Currently, we restrict our attention to filters that are composed of a single constraint, i.e., $f = \phi$. We leave the discussion of filters consisting of multiple constraints and filters corresponding to composite events to Section 4.

Definition 2.2. *Given a publisher, a network and a subscriber shown in Figure 1, a PRIEST scheme in a content-based system consists of*

(1) A service, $\mathbf{S} : \mathcal{T} \times \{0,1\}^* \times \mathcal{O} \times \{0,1\}^* \mapsto \{0,1\}^*$*, for the subscriber to encrypt filter* f *as* $\mathbf{S}(f) = F$*;*

(2) A service, $\mathbf{P} : \mathcal{T} \times \{0,1\}^* \times \{0,1\}^* \mapsto \{0,1\}^*$*, for the publisher to encrypt an event* e *as* $\mathbf{P}(e) = E$*;*

(3) A service, $\mathbf{R} : \{0,1\}^* \mapsto \mathcal{T} \times \{0,1\}^* \times \{0,1\}^*$*, for the subscriber to recover the ciphertext, i.e.,* $\mathbf{R}(\mathbf{P}(e)) = e$.

These functions should satisfy

Correctness: *Event* e *is received by the subscriber if and only if* $\mathbf{R}(\mathbf{P}(e)) \prec_S^N f$*. Namely, a subscriber receives all events that match the filter* f*.*

Privacy: *(1) Given an encrypted attribute, it is computationally infeasible for the network to infer the attribute type, attribute name and attribute value; (2) Given an encrypted constraint, it is computationally infeasible for the network to infer the constraint type, constraint name, constraint value and operator.*

Ideally, it is required that all components of a constraint are kept secret from the network. However, as will be shown later in Section 3.3, it is impossible to hide all information in a filter from the network. More specifically, the operator in a constraint is exposed to the network.

3 PRIEST schemes

In this section, we first show that the PRIEST problem is at least as hard as the single-database private information retrieval problem introduced in [7]. We then describe a PRIEST scheme for a channel-based pub/sub system based on simple XOR operations and one-way collision-resistant hash functions. For a content-based system, we explain why it is impossible to keep all four components of a constraint secret from the network, followed by a PRIEST scheme that discloses the operator in a constraint to the network.

3.1 PRIEST schemes and PIR schemes

The Private Information Retrieval (PIR) problem was introduced in [7] and subsequently received extensive study [3, 5, 6, 9, 15, 16, 25]. In a PIR scheme, a database $\mathcal{B}$ is modelled as an n-bit sequence, namely $\{0,1\}^n$. The PIR scheme enables a user to retrieve the i-th ($1 \leq i \leq n$) bit from $\mathcal{B}$ without revealing i. Chor *et. al.* proved in [7] that, to achieve a single-database PIR scheme that is *unconditionally secure*, the most efficient approach has the same communication complexity as the trivial one in which the entire database is sent to the user, i.e., the communication complexity is $O(n)$. However, to achieve a single-database PIR scheme that is *computationally secure*, there exists a scheme in [5] with $O(n^\epsilon)$ communication complexity for any $\epsilon > 0$.

Theorem 3.1. *PIR$\leq_P$PRIEST, i.e., the single-database PIR problem polytime reduces to the PRIEST problem.* **Proof sketch:** Our proof is to construct a single-database PIR scheme based on a PRIEST scheme in a channel-based system. Assume that there are a total of n channels, indexed as $1, \ldots, n$. The publisher generates either a 1 or a 0 in each channel. The subscriber is interested in the i-th channel. Consider the case when the publisher sent n messages to the network, each for a unique channel. These n messages can be viewed as a database $\mathcal{B}$ composed of an n-bit sequence, i.e., $\mathcal{B} = \{0,1\}^n$, where the j-th bit is the content (either 0 or 1) of the message for the j-th channel. Now a PRIEST scheme enables the network to correctly deliver the message in the i-th channel to the subscriber, namely the i-th bit of $\mathcal{B}$. As a consequence, the subscriber correctly retrieves the i-th bit of $\mathcal{B}$ while keeping i secret from the network because of the first privacy requirement in Definition 2.1. That is, we construct a single-database PIR scheme. ■

The theorem implies that achieving a PRIEST scheme is at least as hard as achieving a single-database PIR scheme. This means that, to achieve an unconditionally-secure PRIEST scheme, the most efficient approach has the same communication complexity as the trivial one in which the entire database is sent to the subscriber, i.e., the subscriber receives all events generated by the publisher. However, we can construct a computationally-secure PRIEST scheme that is more efficient, based on the intractability assumptions described in Section 2.

It is also worth noting that a PRIEST scheme differs from a single-database PIR scheme. In a PIR scheme, for each user's query, the database has to send responses to the user, based on which the user constructs the queried bit. However, in a PRIEST scheme, the network sends an event to the subscriber only when the event matches the filter. Another difference exists; In a PIR scheme, a user is assumed to know the *physical address* of the sought item in the database [6], whereas, most of the interests in PRIEST schemes are specified through content matching, e.g., keywords, values.

3.2 A channel-based PRIEST scheme

A PRIEST scheme, $\mathcal{P}_c$, for a channel-based system consists of the following six steps.

Step 1: The publisher distributes secret keys k and k' to the subscriber. This can be achieved by encrypting k and k' using the subscriber's public key.

Step 2: The publisher secretly delivers all available channel IDs to the subscriber. This can be achieved using Luby-Rackoff's algorithm [20] that is based on three different one-way hash functions and key k';

Step 3: The subscriber chooses a set of IDs to subscribe to, encrypting each chosen ID d as $\mathbf{H}(d \oplus k)$. The encrypted IDs are sent as the subscriber's interest to the network;

Step 4: The publisher encrypts a message $m = (d, x)$ as $M = (\mathbf{H}(d \oplus k), \mathbf{E}_{k'}(x))$, where $\mathbf{E}_{k'}(x)$ is the result of applying Luby-Rackoff's algorithm on x using k';

Step 5: When the network receives M, it checks whether M matches subscriber's interest by XORing $\mathbf{H}(d \oplus k)$ to each encrypted ID received from the subscriber. M matches the interest if and only if the XOR-result of $\mathbf{H}(d \oplus k)$ to one of the encrypted IDs is the all-0 sequence, i.e., $\mathbf{H}(d \oplus k)$ is identical to one of the encrypted IDs. Matched messages are further delivered to the subscriber;

Step 6: After the subscriber receives M from the network, it decrypts the message content using k'.

3.2.1 Analysis

Correctness It is reasonable to assume that the number of predefined channels is less than 2^{n_h}, i.e., there is no collision of $\mathbf{H}(d \oplus k)$. As a consequence, the XORed result of the $\mathbf{H}(d \oplus k)$ in M to one of the encrypted IDs in the subscriber's interest is the all-0 sequence if and only if d matches a channel targeted by the subscriber. That is, the subscriber correctly receives all messages associated with the channels it has targeted. Once the subscriber receives an encrypted message, it recovers the plaintext of the content using k'.

Privacy The computational security of **H** directly provides the computational privacy of $\mathcal{P}_c$. Note that, to thwart brute-force attacks, a plaintext is XORed with a key before it is hashed.

Length of k and k' The length of k and k' can be arbitrary. But for sufficient security, both k and k' should be long enough, e.g., 128 bits.

3.3 A content-based PRIEST scheme

In content-based systems, a subscriber specifies a single constraint $\phi = (type_\phi, name_\phi, operator_\phi, value_\phi)$ in a filter f. Such a filter is encrypted and sent to the network. Every event e generated by the publisher is also encrypted and sent to the network. A PRIEST scheme needs to ensure that the network learns nothing about $type_\phi$, $name_\phi$ and $value_\phi$ while correctly detecting an event e if $e \prec_S^N f$. It is impossible to keep $operator_\phi$ secret from the network because of the following proposition.

Proposition 3.2. *Given a value constraint $\phi = (type_\phi,\ name_\phi,\ operator_\phi,\ value_\phi)$, it is impossible to keep both $operator_\phi$ and $value_\phi$ secret from the network in a PRIEST scheme defined in Definition 2.2.*

Proof sketch: Assume that the network receives an encrypted event $E = \mathbf{P}(e)$. For an attribute $\alpha = (type_\alpha, name_\alpha, value_\alpha)$ of e and the constraint $\phi = (type_\phi,\ name_\phi,\ operator_\phi,\ value_\phi)$, keeping both $operator_\phi$ and $value_\phi$ secret from the network means that Pr[$operator_\phi(value_\alpha, value_\phi)$] equals Pr[$\neg operator_\phi(value_\alpha, value_\phi)$]. Namely, $operator_\phi(value_\alpha, value_\phi)$ and $\neg operator_\phi\ (value_\alpha, value_\phi)$ are equally likely to be true. However, to achieve the correctness requirement, the network must have Pr[$opera\ tor_\phi\ (value_\alpha,\ value_\phi)$] $\geq 1 - c^{-d}$ or Pr[$\neg operator_\phi(value_\alpha, value_\phi)$] $\geq 1 - c^{-d}$, for all constants d and all sufficiently large c. That is a contradiction. ■

The operators in a subscription filter provided by SIENA [4] include all the common equality and ordering relations ($=, \neq, <, >$, etc.) for all of its types; substring ($*$), prefix ($> *$), and suffix ($* <$) operators for strings; and an operator any that matches any value. In this subsection, we describe a PRIEST scheme that supports equality and ordering relations for values. A PRIEST scheme supporting string operators is presented in Section 3.4. Note that a content-based filter with the any operator is similar to an interest in a channel-based system, in which case, the subscriber receives all events that have an attribute whose type and name match the filter.

3.3.1 A PRIEST scheme supporting value filters

Observe that if u, v and r are random numbers, for any equality and ordering operator g, we have $g(u+r, v+r) = g(u,v)$, e.g., $(u+r) \geq (v+r) \Leftrightarrow u \geq v$. In other words, the ordering relation between $u+r$ and $v+r$ is the same as the one between u and v. This means we can "mask" two numbers, u and v, by adding a secret number, r, while allowing the third party to tell the ordering relation between the two numbers and keeping the two numbers secret from the third party. This observation results in a PRIEST scheme, $\mathcal{P}_v$, consisting of the following six steps.

Step 1: The publisher distributes a secret key k and a random number r to the subscriber. This step is similar to the first step of $\mathcal{P}_c$ described in Section 3.2;

Step 2: The publisher secretly announces all possible attribute types and names to the subscriber. The publisher defines the range[2] of each attribute value. Each range is masked by adding r and announced to the subscriber.

Step 3: The subscriber specifies a constraint as $\phi = (type_\phi, name_\phi, operator_\phi, value_\phi)$. ϕ is encrypted as $\Phi = (\mathbf{H}(type_\phi \oplus k), \mathbf{H}(name_\phi \oplus k), operator_\phi, value_\phi + r)$ and sent to the network;

Step 4: The publisher encrypts any attribute $\alpha = (type_\alpha, name_\alpha, value_\alpha)$ of an event e into A=($\mathbf{H}(type_\alpha \oplus k)$, $\mathbf{H}(name_\alpha \oplus k)$, $value_\alpha + r)$. Event $e = \{\alpha_1, \ldots, \alpha_n\}$ consequently turns to be the cipher form $E = \{A_1, \ldots, A_n\}$;

Step 5: When the network receives an encrypted event E, it XORs the $\mathbf{H}(type_\alpha \oplus k)$ and $\mathbf{H}(name_\alpha \oplus k)$ of each A in E with the $\mathbf{H}(type_\phi \oplus k)$ and $\mathbf{H}(name_\phi \oplus k)$ in Φ, respectively. If both produce all-0 sequences, attribute A matches the constraint in type and name. In this case, the network further checks the order relation of the $value_\alpha + r$ of that attribute and the $value_\phi + r$ of Φ by performing $(value_\alpha + r) - (value_\phi + r)$. The difference is compared to the $operator_\phi$ of Φ. All matched events are then delivered to the subscriber;

Step 6: After the subscriber receives an event E from the network, it decrypts the event using k and r.

[2]If a range is not specified, the range can be arbitrary large.

3.3.2 Analysis

Correctness Similar to the correctness of $\mathcal{P}_c$, the network is able to correctly detect an encrypted attribute A matching the encrypted constraint Φ in both attribute types and names. Further, based on $operator_\phi$ and the subtraction of $value_\alpha+r$ and $value_\phi+r$, the network is able to check whether $operator_\phi(value_\alpha, value_\phi)$ is true. As a consequence, the subscriber indeed receives all events matching the filter.

Privacy We show here that encrypting attribute values by adding a random number suffices to ensure computational privacy. Let v and r be a value and the random number respectively and let $u = v + r$. Consider the binary addition of v and r. Let u_i, v_i and r_i be the i-th bit of the binary representation of u, v and r respectively, with the first bit being the least significant bit. Let c_i be the carry bit that is to be added with v_i and r_i (e.g., Pr$[c_1 = 0]$=1). Since r is a random number, we have Pr$[r_i = 1]$=Pr$[r_i = 0]$=1/2. Consequently, we have

$$\begin{aligned}
\Pr[u_i = 1] &= \Pr[v_i = 1]\Pr[r_i = 1]\Pr[c_i = 1] + \Pr[v_i = 0]\Pr[r_i = 0]\Pr[c_i = 1] \\
&\quad +\Pr[v_i = 0]\Pr[r_i = 1]\Pr[c_i = 0] + \Pr[v_i = 1]\Pr[r_i = 0]\Pr[c_i = 0] \\
&= \frac{1}{2}\Pr[c_i = 1](\Pr[v_i = 1] + \Pr[v_i = 0]) + \frac{1}{2}\Pr[c_i = 0](\Pr[v_i = 1] + \Pr[v_i = 0]) \\
&= \frac{1}{2}\Pr[c_i = 1] + \frac{1}{2}\Pr[c_i = 0] = \frac{1}{2}
\end{aligned}$$

Similarly, we have Pr$[u_i = 0]$=1/2. That is, given $u = v + r$, the network can do nothing significantly better than "guessing" v and r.

Length of r The length of an attribute value is defined as the number of bits required to represent the upper range of the value. If there is no range for an attribute value, its length is infinite. The length l_r of r can be chosen as the maximum length of all attribute values that have finite lengths. For a value that has a length longer than l_r, we divide the binary representation of the value into blocks of length l_r. r is then added to each block with the most significant carry bit of a block being considered in the next block.

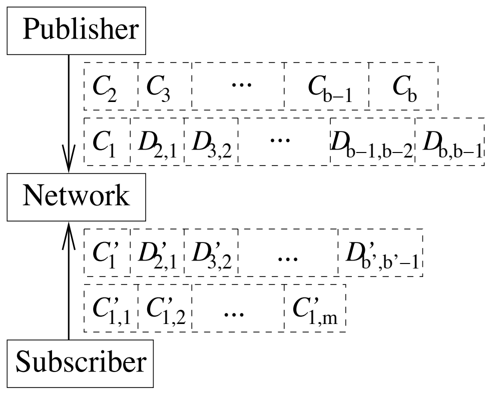
3.4 A PRIEST scheme supporting string filters

In this subsection, we describe a PRIEST scheme for the three string operators provided in SIENA [4]: substring ($*$), prefix ($> *$), and suffix ($* <$). Note that a prefix is a substring in the first position and a suffix is a substring in the last position.

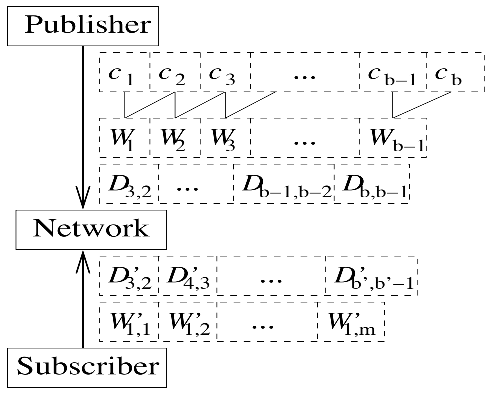
We assume that all strings are composed of the characters with ASCII representations. The three string operators above are all character-unit operators, which means that checking whether two strings are identical corresponds to comparing those two strings character-by-character. As a consequence, a PRIEST scheme supporting string operators should enable the network to perform character-by-character comparison while keeping characters secret from the network. However, a character-by-character comparison is vulnerable to frequency-based attacks. Let $S = C_1C_2 \ldots C_b$ and $S' = C'_1C'_2 \ldots C'_{b'}$ be the encrypted string sent by the publisher and the encrypted substring sent by the subscriber, respectively, where C_i (resp. C'_i) is the ciphertext of character c_i (resp. c'_i). By performing character-by-character comparison, the network is able to detect whether C_i matches C'_j for all $1 \leq i \leq b$ and $1 \leq j \leq b'$. Consequently, the network can classify all encrypted characters into groups such that all characters in a group match each other. For example, all encrypted characters matching C'_1 are classified in the same group. The numbers of characters in different groups reflect the character frequency, based on which the network can infer a plaintext character even though it learns nothing about the encryption. Note that such a frequency-based attack works for any character-unit encryption.

To thwart such attacks, we construct the following scheme. We begin by assuming that all substrings consist of at least two characters. At the end of this subsection, we will describe a PRIEST scheme for single character substrings. The *distance* between two characters is defined as the difference of their corresponding ASCII codes. Namely, the distance d_{ij} between two characters c_i and c_j is $d_{i,j} = a_i - a_j$ mod 128, where a_i and a_j are the ASCII codes of c_i and c_j, respectively. In the rest of this section, we write $d_{i,j} = c_i - c_j$ for notational simplicity. The *distance representation* of a string $s : c_1c_2 \cdots c_b$ is denoted as $\overline{s}$, i.e., $\overline{s} = c_1d_{2,1}d_{3,2} \cdots d_{b,b-1}$. Clearly, string $\overline{s} : c_1d_{2,1}d_{3,2} \cdots d_{b,b-1}$ equals string $\overline{s'} : c'_1d'_{2,1}d'_{3,2} \cdots d'_{b,b-1}$ if and only if $c_1 = c'_1$ and $d_{i,i-1} = d'_{i,i-1}$ for all $2 \leq i \leq b$.

Assume that $\overline{s} : c_1d_{2,1}d_{3,2} \cdots d_{b,b-1}$ is encrypted as $\overline{S} : C_1D_{2,1}D_{3,2} \cdots D_{b,b-1}$, and $\overline{s'} : c'_1d'_{2,1}d'_{3,2} \cdots d'_{b',b'-1}$ is encrypted as $\overline{S'} : C'_1D'_{2,1}D'_{3,2} \cdots D'_{b',b'-1}$. Since the first character c'_1 of s' can be the i_0-th character of s for $1 \leq i_0 \leq b - b' + 1$, s includes s' as a substring if and only if there exists an i_0 such that C'_1 equals C_{i_0}, the ciphertext of character c_{i_0}, and $D'_{i+2,i+1} = D_{i_0+i+1,i_0+i}$ for all $0 \leq i \leq b' - 2$. Given only $\overline{S}$ and $\overline{S'}$, it is impossible for the network to determine whether s' is a substring of s since C_i ($2 \leq i \leq b$) is not available to the network. Thus the ciphertext $S = C_1C_2 \cdots C_b$ of s must be provided for the network to determine whether s includes s'. However, if the same character in s is encrypted as the same ciphertext, the network can learn the character frequency by grouping the ciphertext characters of S, i.e., two ciphertext characters in S are classified in a group if their ciphertext is identical. This problem can be solved using probabilistic encryption [13]. More specifically, a character in s is encrypted as different ciphertext using different keys. Let $\mathcal{K} = \{k_1, \ldots, k_m\}$ be the set of keys to encrypt characters. For each character c of s, the publisher randomly chooses a key $k_i \in \mathcal{K}$ and encrypts c using k_i. Thus a character can be encrypted as m possible distinct ciphertext, which conceals the character frequency from the network. With probabilistic encryption, the first character c'_1 of s' can be encrypted by the publisher as a total of m possible ciphertext. Correspondingly, the subscriber needs to send the m ciphertext of c'_1, denoted as $C'_{1,1}, \ldots, C'_{1,m}$, to ensure that the network is able to recognize all ciphertext of c'_1 in S. This approach is shown in Figure 2(a).


Publisher
C_2 C_3 ... C_{b-1} C_b
C_1 $D_{2,1}$ $D_{3,2}$ ... $D_{b-1,b-2}$ $D_{b,b-1}$
Network
C'_1 $D'_{2,1}$ $D'_{3,2}$ ... $D'_{b',b'-1}$
$C'_{1,1}$ $C'_{1,2}$ ... $C'_{1,m}$
Subscriber

(a)


Publisher
c_1 c_2 c_3 ... c_{b-1} c_b
W_1 W_2 W_3 ... W_{b-1}
$D_{3,2}$ ... $D_{b-1,b-2}$ $D_{b,b-1}$
Network
$D'_{3,2}$ $D'_{4,3}$ ... $D'_{b',b'-1}$
$W'_{1,1}$ $W'_{1,2}$ ... $W'_{1,m}$
Subscriber

(b)

Figure 2: Two approaches concealing character frequency.

The approach in Figure 2(a) still reveals the character frequency of the first character c'_1 of s', namely, the ciphertext characters in S matching an element in $\{C'_{1,1}, C'_{1,2}, \ldots, C'_{1,m}\}$ are classified into a group. A further improvement completely prevents the network from learning the frequency of any single character. Figure 2(b) shows the improved approach. Two adjacent characters in string s are encrypted as a ciphertext word, i.e., $c_i c_{i+1}$ is encrypted as W_i ($1 \le i \le b-1$) using a key randomly chosen from $\mathcal{K}$. The publisher deletes the first character c_1 and the first distance $d_{2,1}$ from $\overline{s}$ and encrypts the rest as $D_{3,2}D_{4,3} \cdots D_{b,b-1}$. Correspondingly, the subscriber encrypts $d'_{3,2}d'_{4,3} \cdots d'_{b',b'-1}$ as $D'_{3,2}D'_{4,3} \cdots D'_{b',b'-1}$. It also encrypts the first two characters $c'_1c'_2$ of s' as a total of m ciphertext, denoted as $W'_{1,1}, \ldots, W'_{1,m}$, using m keys in $\mathcal{K}$.

Based on the approach in Figure 2(b), we construct a PRIEST scheme, $\mathcal{P}_s$, consisting of the following six steps.

Step 1: The publisher distributes to the subscriber a key k, a set $\mathcal{R} = \{r_1, r_2, \ldots, r_m\}$ of random numbers, and a key set $\mathcal{K} = \{k_1, k_2, \ldots, k_m\}$, where $k \notin \mathcal{K}$ is used to encrypt attribute types and names, $\mathcal{R}$ is used to encrypt character distances, and $\mathcal{K}$ is used to encrypt characters. Note that in practice, $\mathcal{R}$ and $\mathcal{K}$

may have different sizes;

Step 2: The publisher secretly distributes all possible attribute types and names to the subscriber;

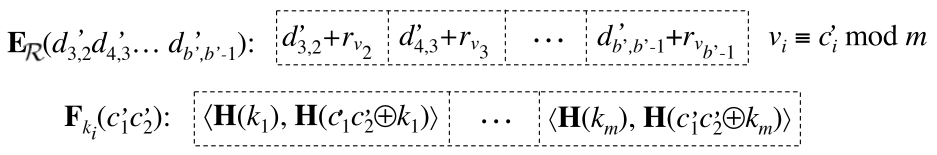
Step 3: The subscriber specifies a constraint $\phi = (type_\phi,\ name_\phi,\ operator_\phi,\ s_\phi : c'_1c'_2 \cdots c'_{b'})$. When $operator_\phi$ is "$*$" (resp. "$>*$" and "$*<$"), the subscriber is interested in strings that include (resp. begin with and end with) s_ϕ. Let $c'_1d'_{2,1} \cdots d'_{b',b'-1}$ be the distance representation of s_ϕ. ϕ is encrypted as $\Phi = (\mathbf{H}(type_\phi \oplus k),\ \mathbf{H}(name_\phi \oplus k),\ operator_\phi,\ \mathbf{E}_\mathcal{R}(d'_{3,2}d'_{4,3} \cdots d'_{b',b'-1}), \mathbf{F}_{k_1}(c'_1c'_2),\ \mathbf{F}_{k_2}(c'_1c'_2),$

$\ldots, \mathbf{F}_k$ n_h, and

$\mathbf{E}_\mathcal{R}(d'_3$. More

specific aracter

c'_{i-1}. T



$\mathbf{E}_\mathcal{R}(d'_{3,2}d'_{4,3} \ldots d'_{b',b'-1})$:
$d'_{3,2}+r_{v_2}$
$d'_{4,3}+r_{v_3}$
$\cdots$
$d'_{b',b'-1}+r_{v_{b'-1}}$
$v_i \equiv c'_i \bmod m$
$\mathbf{F}_{k_i}(c'_1c'_2)$:
$\langle \mathbf{H}(k_1), \mathbf{H}(c'_1c'_2 \oplus k_1)\rangle$
$\cdots$
$\langle \mathbf{H}(k_m), \mathbf{H}(c'_1c'_2 \oplus k_m)\rangle$

Figure 3: Ciphertext of $s_\phi = c'_1c'_2 \cdots c'_{b'}$ sent by the subscriber.

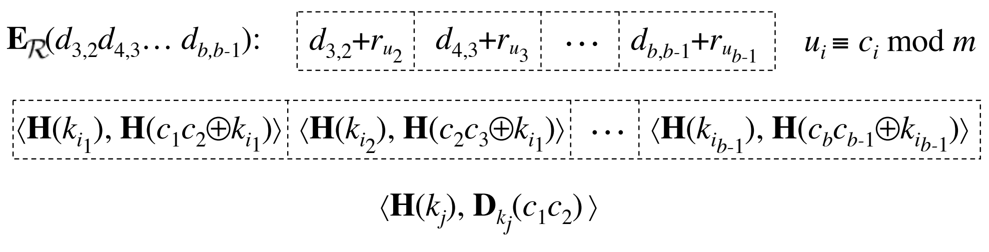
Step 4: The publisher encrypts a string attribute $\alpha = (type_\alpha, name_\alpha, s_\alpha : c_1c_2 \cdots c_b)$ as A=$(\mathbf{H}(type_\alpha \oplus$ $k), \mathbf{H}(name_\alpha \oplus k), \mathbf{E}_\mathcal{R}(d_{3,2}d_{4,3} \cdots d_{b,b-1}), \mathbf{F}_{k_{i_1}}(c_1c_2), \mathbf{F}_{k_{i_2}}(c_2c_3), \ldots, \mathbf{F}_{k_{i_{b-1}}}(c_{b-1}c_b), \langle \mathbf{H}(k_j), \mathbf{D}_{k_j}(c_1c_2)\rangle)$,

whe en-

cry $_1c_2$

and orm

E =



$\mathbf{E}_\mathcal{R}(d_{3,2}d_{4,3} \ldots d_{b,b-1})$:
$d_{3,2}+r_{u_2}$
$d_{4,3}+r_{u_3}$
$\cdots$
$d_{b,b-1}+r_{u_{b-1}}$
$u_i \equiv c_i \bmod m$
$\langle \mathbf{H}(k_{i_1}), \mathbf{H}(c_1c_2 \oplus k_{i_1})\rangle$
$\langle \mathbf{H}(k_{i_2}), \mathbf{H}(c_2c_3 \oplus k_{i_1})\rangle$
$\cdots$
$\langle \mathbf{H}(k_{i_{b-1}}), \mathbf{H}(c_bc_{b-1} \oplus k_{i_{b-1}})\rangle$
$\langle \mathbf{H}(k_j), \mathbf{D}_{k_j}(c_1c_2)\rangle$

Figure 4: Ciphertext of $s_\alpha = c_1c_2 \cdots c_b$ sent by the publisher.

Step 5: Given an attribute A=$(\mathbf{H}(type_\alpha \oplus k),\ \mathbf{H}(name_\alpha \oplus k),\ \mathbf{E}_\mathcal{R}(d_{3,2}d_{4,3} \cdots d_{b,b-1}),\ \mathbf{F}_{k_{i_1}}(c_1c_2),$

$\mathbf{F}_{k_{i_2}}(c_2c_3)$, …, $\mathbf{F}_{k_{i_{b-1}}}(c_{b-1}c_b)$, $\langle \mathbf{H}(k_j), \mathbf{D}_{k_j}(c_1c_2)\rangle$) and a constraint $\Phi = (\mathbf{H}(type_\phi \oplus k), \mathbf{H}(name_\phi \oplus k)$, $operator_\phi$, $\mathbf{E}_{\mathcal{R}}(d'_{3,2}d'_{4,3}\cdots d'_{b',b'-1})$, $\mathbf{F}_{k_1}(c'_1c'_2)$, $\mathbf{F}_{k_2}(c'_1c'_2)$, …, $\mathbf{F}_{k_m}(c'_1c'_2))$ such that $\mathbf{H}(type_\alpha \oplus k) = \mathbf{H}(type_\phi \oplus k)$ and $\mathbf{H}(name_\alpha \oplus k) = \mathbf{H}(name_\phi \oplus k)$, the network performs the following operations depending on $operator_\phi$:

(i). $operator_\phi$=" $*$ ":

```
For each $l = 1, \ldots, b - b' + 1$
  If there exists a $1 \leq j \leq m$ such that $\mathbf{F}_{k_j}(c'_1c'_2) = \mathbf{F}_{k_{i_l}}(c_lc_{l+1})$
    Compare $d'_{t,t-1} + r_{v_{t-1}}$ with $d_{t+l-1,t+l-2} + r_{u_{t+l-2}}$ for $t = 3, 4, \ldots, b'$.
    If all these $b' - 2$ encrypted distances are identical, a " $*$ " matching is detected.
```

Figure 5: Operations to detect a " $*$ " matching

(ii). $operator_\phi$=" $> *$": The operation for a suffix operator is a special case of the operation for a substring operator when $l = 1$. Namely:

```
If there exists a $1 \leq j \leq m$ such that $\mathbf{F}_{k_j}(c'_1c'_2) = \mathbf{F}_{k_{i_1}}(c_1c_2)$
  Compare $d'_{t,t-1} + r_{v_{t-1}}$ with $d_{t,t-1} + r_{u_{t-1}}$ for $t = 3, 4, \ldots, b'$.
  If all these $b' - 2$ encrypted distances are identical, a " $> *$" matching is detected.
```

Figure 6: Operations to detect a " $> *$" matching

(iii). $operator_\phi$="$* <$ ": The operation for a suffix operator is a special case of the operation for a substring operator when $l = b - b' + 1$. Namely:

```
If there exists a $1 \leq j \leq m$ such that $\mathbf{F}_{k_j}(c'_1c'_2) = \mathbf{F}_{k_{i_{b-b'+1}}}(c_{b-b'+1}c_{b-b'+2})$
  Compare $d'_{t,t-1} + r_{v_{t-1}}$ with $d_{t+b-b',t+b-b'-1} + r_{u_{t+b-b'-1}}$ for $t = 3, 4, \ldots, b'$.
  If all these $b' - 2$ encrypted distances are identical, a " $> *$" matching is detected.
```

Figure 7: Operations to detect a "$* <$ " matching

All matched events are then delivered to the subscriber;

Step 6: After the subscriber receives a matched attribute from the network, it performs the following operations to recover string s_α. Given $\langle \mathbf{H}(k_j), \mathbf{D}_{k_j}(c_1c_2)\rangle$, the subscriber is able to separate $\mathbf{H}(k_j)$ from $\mathbf{D}_{k_j}(c_1c_2)$ since the hashed value of $\mathbf{H}$ is of length n_h. The network determines k_j by computing $\mathbf{H}(k_i)$ for each $k_i \in \mathcal{K}$. If $\mathbf{H}(k_i)$ and $\mathbf{H}(k_j)$ are identical, k_i equals k_j and consequently, the subscriber recovers c_1c_2 by decrypting $\mathbf{D}_{k_j}(c_1c_2)$ using k_i. Based on c_2, the subscriber calculates $u_2 \equiv c_2 \bmod m$ and obtains $d_{3,2}$

by subtracting r_{u_2} from $d_{3,2} + r_{u_2}$, which further reveals c_3. Similarly, the rest of the characters in s_α are recovered.

3.4.1 Analysis

Correctness The correctness of $\mathcal{P}_s$ can be verified through the following statements.

- $c'_1 c'_2 = c_l c_{l+1} \Leftrightarrow \exists j : \mathbf{F}_{k_j}(c'_1 c'_2) = \mathbf{F}_{k_{i_l}}(c_l c_{l+1})$. Recall that $\mathbf{F}_{k_{i_l}}(c_l c_{l+1}) = \langle \mathbf{H}(k_{i_l}), \mathbf{H}(c_l c_{l+1} \oplus k_{i_l})\rangle$. The collision resistance property of $\mathbf{H}$ guarantees that if $k_j \neq k_{i_l}$, the first n_h bits of $\mathbf{F}_{k_j}(c'_1 c'_2)$ and $\mathbf{F}_{k_{i_l}}(c_l c_{l+1})$ are different[3]. So $\mathbf{F}_{k_j}(c'_1 c'_2) = \mathbf{F}_{k_{i_l}}(c_l c_{l+1}) \Rightarrow \exists j : k_j = k_{i_l}$. Since there are total 128×128 different combinations of two characters, it is easy to ensure that, for key k_{i_l}, $\mathbf{H}(c_l c_{l+1} \oplus k_{i_l})$ is distinct for different $c_l c_{l+1}$. Consequently, when k_j equals k_{i_l}, the last n_h bits of $\mathbf{F}_{k_j}(c'_1 c'_2)$ and $\mathbf{F}_{k_{i_l}}(c_l c_{l+1})$ are identical if and only if $c'_1 c'_2 \oplus k_j = c_l c_{l+1} \oplus k_{i_l}$, i.e., $c'_1 c'_2 = c_l c_{l+1}$.
- $(c'_1 c'_2 = c_l c_{l+1}) \wedge (\forall t \in [3, b'] : d'_{t,t-1} + r_{v_{t-1}} = d_{t+l-1,t+l-2} + r_{u_{t+l-2}}) \Leftrightarrow s_\alpha$ includes s_ϕ. Since $c'_2 = c_{l+1}$, we have $v_2 = u_{l+1} \Leftrightarrow r_{v_2} = r_{u_{l+1}} \Leftrightarrow d'_{3,2} = d_{l+2,l+1} \Leftrightarrow c'_3 = c_{l+2}$. Similarly, we have $c'_i = c_{l+i-1}$ for $i = 4, \ldots, b'$. As a result, s_ϕ is a substring of s_α.
- s_α is recovered correctly by the subscriber. This can be easily verified, as described in Step 6.

Privacy Most of the information that the network receives are hashed values of the one-way function $\mathbf{H}$. The privacy of these information is achieved based on the security of $\mathbf{H}$. The privacy of $\mathbf{E}_{\mathcal{R}}(d_{3,2} d_{4,3} \cdots d_{b,b-1})$ can be shown through a similar proof to the privacy of the encrypted values in PRIEST scheme $\mathcal{P}_v$ described in Section 3.3.1.

It should be emphasized that $\mathcal{P}_s$ is reasonably secure against a frequency-based attack since the frequency of no single character is revealed. This is achieved by using probabilistic encryption with a total of m different encryption keys.

We remark that, similar to $\mathcal{P}_v$, operators in $\mathcal{P}_s$ have to be made known to the network.

3.4.2 Single-character substrings

We briefly describe a scheme $\mathcal{P}'_s$ that handles the case where a subscriber's constraint consists of a single character. $\mathcal{P}'_s$ is constructed based on the intractability assumption of the FACTORING problem. $\mathcal{P}'_s$ works

[3]In practice, $m = 10$ is enough to conceal the character frequency and it is easy to find ten keys that have distinct hashed values.

as follows. All characters are mapped to pairwise distinct primes by a mapping agreed to by the publisher and subscriber. Let p_c be the prime corresponding to character c. Let $\mathcal{C}$ be the set of characters appearing in string $s : c_1c_2 \cdots c_n$. The publisher generates $N = (ap_{c_1})^2(bp_{c_n})^2 \prod_{c_i \in \mathcal{C} \setminus \{c_1,c_n\}} p_{c_i}$, where a and b satisfy the following three conditions[4]: (1) $a \neq b$; (2) gcd(a, b)=1; and (3) $a, b \neq p_c$ for all $c \in \mathcal{C}$. Let c be the character that the subscriber is interested in; c is encrypted in different forms depending on $operator_\phi$. If $operator_\phi =$ " $> *$", c is encrypted as $C = (ap_c)^2$. If $operator_\phi =$ " $*$ ", c is encrypted as $C = p_c^2$. If $operator_\phi =$ "$* <$ ", c is encrypted as $C = (bp_c)^2$. When the network receives N and C, it performs different operations depending on $operator_\phi$. If $operator_\phi =$ " $> *$", the network determines whether s has a prefix of c by checking whether C divides N, since $C \mid N \Leftrightarrow (ap_c)^2 \mid (ap_{c_1})^2(bp_{c_n})^2 \prod_{c_i \in \mathcal{C} \setminus \{c_1,c_n\}} p_{c_i} \Leftrightarrow p_c = p_{c_1}$. Similarly, if $operator_\phi =$ "$* <$ ", the network checks whether C divides N since $C \mid N \Leftrightarrow (bp_c)^2 \mid (ap_{c_1})^2(bp_{c_n})^2 \prod_{c_i \in \mathcal{C} \setminus \{c_1,c_n\}} p_{c_i} \Leftrightarrow p_c = p_{c_n}$. If $operator_\phi =$ " $*$ ", the network checks whether C divides N^2 since $C \mid N^2 \Leftrightarrow (p_c)^2 \mid (ap_{c_1})^4(bp_{c_b})^4 \prod_{c_i \in \mathcal{C} \setminus \{c_1,c_n\}} p_{c_i}^2 \Leftrightarrow \exists c_i \in \mathcal{C} : p_c = p_{c_i}$. In this case, the network only learns C and N. Given C, it is intractable to infer p_c. Similarly, given N, it is intractable to infer any p_{c_i} for $c_i \in \mathcal{C}$. In $\mathcal{P}'_s$, the network does not need to perform a sequence scan on the ciphertext of s. Consequently, we do not need to encrypt s character-by-character. For example, s can be encrypted using Luby-Rackoff's algorithm, which is used in PRIEST scheme $\mathcal{P}_c$ to encrypt message contents in a channel-based system. Note that, since the network cannot learn any p_{c_i} and the length n of s, $\mathcal{P}'_s$ is secure against frequency-based attacks.

This section closes with a word on the order operators (e.g., $=, \neq, <, >$) on strings. When a character is mapped to a value of its ASCII code, these operators on strings can be supported in the $\mathcal{P}_v$ described in Section 3.3.

4 Discussions

In this section, we discuss several additional issues related to the PRIEST problem.

4.1 Multiple-constraint filters

In Section 3, we only considered filters consisting of a single constraint. Since the network is able to detect events matching a single encrypted constraint, the PRIEST scheme described above can be easily extended to detect events matching multiple constraints. However, the relationship among the constraints

[4]In practice, we can choose $a = 2$ and $b = 3$.

(i.e., conjunction or disjunction) needs to be exposed to the network.

4.2 Composite-event filters

Composite events have been introduced in [8, 12, 14, 19, 21] in order to allow subscribers to receive events that satisfy complex patterns. Typically, a composite event is specified in terms of conditions or constraints on attribute values (e.g., timestamp) of a set of primitive events [18]. When the set of primitive events satisfying the appropriate conditions, we say that the composite event has occurred. We assume in this paper that the network (rather than the publisher) will generate a notice to the subscriber at the occurrence of a composite event. The network is required to be able to detect the occurrence of a composite event without learning anything about primitive events.

In the following, we describe how the PRIEST scheme can support composite event filters for the five composition schemes discussed in [8].

Disjunction: The event type $\mathcal{E}$ characterizing the disjunction of two events e_1 and e_2 is of the form $\mathcal{E}_1 \parallel \mathcal{E}_2$. An instance e of $\mathcal{E}$ occurs iff e_1 or e_2 (or both) occurs.

Conjunction: The event type $\mathcal{E}$ characterizing the conjunction of two events e_1 and e_2 is of the form $\mathcal{E}_1\&\&\mathcal{E}_2$. An instance e of $\mathcal{E}$ occurs iff e_1 and e_2 occur regardless of their order of occurrence.

Sequence: The event type $\mathcal{E}$ characterizing the sequence of two events e_1 and e_2 is of the form: $\mathcal{E}_1, \mathcal{E}_2$. An instance of $\mathcal{E}$ occurs iff e_1 occurs before e_2.

Iteration: The event type $\mathcal{E}$ characterizing the iteration of n events of type $\mathcal{E}_1$ is of the form: $n(\mathcal{E}_1)$. An instance of this type occurs iff n events of type $\mathcal{E}_1$ occur in sequence, n is a positive integer.

Negation: The event type $\mathcal{E}$ characterizing the negation of an event e_1 is of the form: $!(\mathcal{E}_1)$. An instance of this type occurs iff e_1 does not occur within a time interval of interest.

Disjunction, conjunction and iteration compositions It is clear that the PRIEST scheme described above supports the disjunction, conjunction, and iteration compositions. Either e_1 or e_2 can be privately detected by the network in the PRIEST scheme. The network only needs to maintain states for occurrences of e_1 and e_2 to detect an event of $\mathcal{E}_1 \parallel \mathcal{E}_2$ or $\mathcal{E}_1\&\&\mathcal{E}_2$. Additionally, the network can introduce a counter for the occurrence of a primitive event to detect an event of $n(\mathcal{E}_1)$.

Sequence composition As described in Section 3, the network is able to detect the order relationship between two values without learning anything about the values themselves. Assume that the subscriber's filter is a sequence composition requiring e_1 to occur before e_2. The subscriber first encrypts the constraints of e_1 and e_2 to be Φ_1 and Φ_2, respectively. The subscriber consequently encrypts the sequence composition filter as $F_S = (\Phi_1, \Phi_2, \mathbf{H}(attribute\_time \oplus k), \Phi_1 < \Phi_2)$. To detect whether two events E_1 (matching Φ_1) and E_2 (matching Φ_2) satisfy F_S, the network first finds the attribute in E_1 and E_2 by XORing the encrypted attribute name with $\mathbf{H}(attribute\_time \oplus k)$ and then compares the order relationship of the associated values in E_1 and E_2.

Negation composition We describe two different compositions that depend on how the time interval, henceforth called validity interval, is defined.

- If the subscriber wants to receive a notice when an event e does not occur in an interval v starting from time t, the subscriber will send the network the ciphertext of the filter $F_\neg = (\Phi, \mathbf{H}(attribute\_time \oplus k), t+v+r, \text{“}negation\text{”})$, where Φ is the encrypted constraint of e[5]. When the network receives $F_\neg$, it maintains a state for the occurrence of e. $F_\neg$ is satisfied if the network receives an encrypted event E whose timestamp is greater than $t + v$ while there is no occurrence of e[6].
- If the subscriber wants to receive a notice when an event e does not occur in an interval v starting from the occurrence of another event e', the subscriber will send the encrypted filter $F'_\neg = (\Phi, \Phi', \mathbf{H}(attribute\_time \oplus k), v, \text{“}negation\text{”})$, where Φ and Φ' are the encrypted constraints of e and e', respectively. When the network detects the occurrence of e', it records the secret timestamp $t' + r$ of e' and maintains a state for the occurrence of e. $F'_\neg$ will be satisfied if the network receives an encrypted event E with a timestamp $t + r$ such that $(t+r) - (t'+r) > v$ while there is no occurrence of e.

Similar to the fact that $operator_\phi$ in each encrypted constraint is exposed to the network, composite event filters expose part of information to the network, e.g., $\Phi_1 < \Phi_2$ in F_S and v in $F'_\neg$.

4.3 Collusion

The first requirement of the PRIEST scheme, that the subscriber hide its interest from the network, prevents any collusion between the subscriber and the network. Similarly, the second requirement, that the publisher

[5]Here we assume that the time is synchronized between the publisher and the subscriber.
[6]Here we also assume that events are sent to the network in the order they are generated.

keep events secret from the network, prevents any collusion between the publisher and the network.

Let us relax the second requirement and only consider the first one. In this case, if the publisher colludes with the network, a PRIEST scheme becomes a PrivatE Retrieval scheme by KeYwords (PERKY) [6], except that a PRIEST scheme requires order relations and string operators.

We should also note that the publisher and the network may exchange any information other than the subscriber's interests, although they cannot collude. For example, the network can report the number of matched events to the publisher so that the publisher can charge the subscriber cost of receiving matched events.

5 Related work

The most related work to this paper is the scheme proposed by Song *et al.* [22], where a customer stores encrypted data on a remote database and wants to search the database by a keyword while keeping the keyword secret to the database. It is clear, however, that the PRIEST problem is more complicated than the problem considered in [22], e.g., the PRIEST problem requires not only keyword matching, but also value ordering detection and string operations. Particularly, the problem considered in [22] corresponds to the channel-based PRIEST scheme $\mathcal{P}_c$, where the channel IDs are the keywords that the subscriber wants to search. From a communication complexity point of view, the number of encrypted channel IDs sent from the publisher to the network in $\mathcal{P}_c$ is the same as the number of the words sent from the customer to the remote database in [22]. Whereas, the number of encrypted filters sent from the subscriber to the network in $\mathcal{P}_c$ is less than the number of encrypted queries sent from the customer to the database in [22]. More specifically, an encrypted query in the latter case is composed of an encrypted keyword and additional information of the key to be used for the database to conduct the search. On the other hand, an encrypted filter in the former case does not include the key information.

Recently, Boneh *et al.* proposed a scheme of Public key Encryption with Keyword Search (PEKS) [2]. A sender encrypts an email with several keywords using a receiver's public key and sends the ciphertext to a mail server. A PEKS scheme enables the mail server, by requiring the receiver to send a trapdoor to the server, to test whether an email consists of a keyword w without learning anything else about the email. A PRIEST scheme is quite similar to a PEKS scheme. Two differences are worth noting though. First, a PEKS scheme only provides word-by-word search whereas a PRIEST scheme supports character-by-character matching. Second, a PRIEST scheme is based on symmetric encryptions whereas a PEKS scheme is based on public-key encryptions. In a pub/sub system, the publisher may generate a large amount of data. In this case, a PEKS scheme is not appropriate for the high encryption overhead of public-key systems.

6 Conclusions

In this paper, we formally introduced the private subscription problem in pub/sub systems, which is shown to be at least as hard as the single-database Private Information Retrieval (PIR) problem introduced in [7]. Since the most efficient approach of the unconditionally-secure PIR problem has the same communication complexity as the trivial one in which the entire database is sent to the user, we described a computationally-secure PRIEST scheme based on the one-wayness assumption of the hash functions and the intractability assumption of the integer factorization problem. The proposed PRIEST scheme combines information confidentiality and subscription confidentiality against the network to achieve a strong level of user privacy. Our scheme is flexible that it supports both channel-based and content-based pub/sub systems, the latter which supports both primitive-event detections and composite-event detections. To that end, the PRIEST scheme is also efficient and practical to use in current pub/sub systems wherein a large amount of data may be generated.

References

[1] P. van Oorschot A. Menezes and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[2] D. Boneh, G. Crescenzo, R. Ostrovsky, and G. Persiano. Public-key encryption with keyword search. In *C. Cachin, editor, Proceedings of Eurocrypt 2004*, Interlaken, Switzerland, May 2004.

[3] C. Cachin, S. Micali, and M. Stadler. Computationally private information retrieval with polylogarithmic communication. *Lecture Notes in Computer Science*, 1592:402–414, 1999.

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

[5] B. Chor and N. Gilboa. Computationally private information retrieval. In *Twenty-ninth annual ACM symposium on Theory of computing*, pages 304–313, El Paso, Texas, May 1997.

[6] B. Chor, N. Gilboa, and M. Naor. Private information retrieval by keywords. Technical Report TR CS0917, Department of Computer Science, Technion, 1997.

[7] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. In *36th IEEE Conference on the Foundations of Computer Science (FOCS)*, pages 41–50, 1995.

[8] C. Collet and T. Coupaye. Primitive and composite events in NAOS. In *Actes des 12e Journees Bases de Donneees Avancees*, pages 331–349, Cassis, France, September, 1996.

[9] G. Crescenzo, Y. Ishai, and R. Ostrovsky. Universal service-providers for database private information retrieval (extended abstract). In *Symposium on Principles of Distributed Computing*, pages 91–100, 1998.

[10] J. Fortes. Transnational digital government research: Building regional partnerships. Highlighted case study presentation at the Digital Government conference, 2003.

[11] Z. Ge, P. Ji, J. Kurose, and D. Towsley. Matchmaker: Signaling for dynamic publish/subscribe applications. In *Proceedings of IEEE ICNP 2003*, Atlanta, GA, November 4-7, 2003.

[12] N. H. Gehani, H. V. Jagadish, and O. Shmueli. Composite event specification in active databases: Model & implementation. In *Proceedings of the 18th International Conference on Very Large Databases*, Vancouver, British Columbia, Canada, 1992.

[13] S. Goldwasser and S. Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *the 18th ACM Symposium on theory of computing*, pages 270–299, 1982.

[14] Annika Hinze. Efficient filtering of composite events. In *Proceedings of the 20th British National Database Conference*, Coventry, UK, 2003.

[15] Y. Ishai and E. Kushilevitz. Improved upper bounds on information-theoretic private information retrieval (extended abstract). In *31th Annu. ACM Symp. on the Theory of Computing (STOC)*, pages 79–88, 1999.

[16] E. Kushilevitz and R. Ostrovsky. Replication is not needed: Single database, computationally-private information retrieval. In *38th IEEE Conference on the Foundations of Computer Science (FOCS)*, Miami Beach, Florida, October 1997.

[17] E. M. Lee, S. Su, and H. Lam. Event and rule services for achieving a web-based knowledge network. In *Web Intelligence 2001*, pages 205–216, Maebashi, Japan, October, 2001.

[18] G. Liu and A. Mok. Implementation of JEM - a java composite event package. In *Proceedings of Fifth IEEE Real-Time Technology and Applications Symposium*, Vancouver, Canada, June, 1999.

[19] G. Liu, A. Mok, and E. Yang. Composite events for network event correlation. In *Proceedings of IFIP/IEEE International Symposium on Integrated Network Management*, Boston, MA, May, 1999.

[20] M. Luby and C. Rackoff. How to construct pseudo-random permutations from pseudorandom functions. *SIAM Journal on Computing*, pages 373–386, April 1988.

[21] P. R. Pietzuch, B. Shand, and J. Bacon. A framework for event composition in distributed systems. In *Proceedings of the 4th International Conference on Middleware (MW'03)*, Rio de Janeiro, Brazil, June, 2003.

[22] D. Song, D. Wagner, and A. Perrig. Practical techniques for searches on encrypted data. In *IEEE Symposium on Security and Privacy*, pages 44–55, Oakland, California, May 2000.

[23] R. Strom, G. Banavar, T. Chandra, M. Kaplan, K. Miller, B. Mukherjee, D. Sturman, and M. Ward. Gryphon: An information flow based approach to message brokering. In *International Symposium on Software Reliability Engineering '98 Fast Abstract*, 1998.

[24] C. Wang, A. Carzaniga, D. Evans, and A. L. Wolf. Security issues and requirements for internet-scale publish- subscribe systems. In *35th Hawaii International Conference on System Science (HICSS-35)*, Hawaii, 2002.

[25] A. Yamamura and T. Saito. Private information retrieval based on the subgroup membership problem. *Lecture Notes in Computer Science*, 2119:206–220, 2001.