Computer Communications 76 (2016) 42–53

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

AnonPubSub: Anonymous publish-subscribe overlays

Jörg Daubert[a,d,*], Mathias Fischer[b], Tim Grube[a], Stefan Schiffner[c], Panayotis Kikiras[d], Max Mühlhäuser[a]

[a] Technische Universität Darmstadt – CASED, Hochschulstr. 10, Darmstadt 64289, Germany
[b] Networking and Security Group, International Computer Science Institute, Berkeley, USA
[c] European Union Agency for Network and Information Security (ENISA), Greece
[d] AGT Group (R&D) GmbH, Hilpertstr. 35, Darmstadt 64295, Germany

ARTICLE INFO

Article history:
Received 17 January 2015
Revised 6 November 2015
Accepted 8 November 2015
Available online 2 December 2015

Keywords:
Anonymity
Overlay networks
Publish-subscribe

ABSTRACT

Publish-subscribe is an increasingly popular messaging pattern for distributed systems, supporting scalable and extensible programming, and optimal spatial, temporal, and control-flow decoupling of distributed components. Publish-subscribe middleware and methods were extended towards supporting security, in particular confidentiality, and increased availability, yet a few prior works addressed anonymity of participants. Anonymity of senders and receivers may however be crucial, e.g., for supporting freedom of expression in regimes where political repression and censorship prevail. In this article, we review basic security and privacy requirements and introduce a new attacker model based on statistical disclosure, used to challenge anonymity. We elaborate on design options for privacy-preserving publish-subscribe systems and present a novel system that leverages peer-to-peer networking concepts; this novel approach protects subscriber anonymity by means of Probabilistic Forwarding (PF) and through a novel so-called Shell Game (SG) algorithm. We verify our solution against the requirements and provide a simulation-based analysis of the effectiveness of our approaches in light of our attacker model. The results show that the SG algorithm efficiently protects subscriber anonymity, and that anonymity sets can be adjusted via PF.

1. Introduction

Publish-subscribe (Pub-sub) drives many application in privacy-challenging environments, e.g., private car sharing systems or in a citizen journalism scenario in which participants could be subject of repression and political prosecution. Such a citizen journalism scenario matches micro-blogging services like Twitter.

A pub-sub system disseminates information from producers (*publishers*) to consumers (*subscribers*) using third parties, so-called *brokers*. Brokers store and match interests of subscribers (*subscriptions*) against *publications* from publishers, to forward the publications as *notifications* to subscribers.

Existing solutions for secure and privacy-preserving pub-sub [1–5] only protect the confidentiality of the exchanged information, so that the content of messages is not leaked to curious attackers. However, these systems do not protect the meta data of the communication such as the network IDs of publishers and subscribers—the anonymity.

A dissident may for instance exchange encrypted information with other dissidents (confidentiality). Communication meta data however still reveals the fact that he communicates with dissidents (no anonymity). This may already lead to political repression.

We therefore argue that a privacy-preserving pub-sub system must protect confidentiality and anonymity as well. In our scenario, once an attacker gets hold of one dissident's encryption keys, other dissidents may be easily tracked down if anonymity is not protected as well. Furthermore, for application scenarios like citizen journalism, content should be made available to a large audience, and therefore does not allow for confidentiality in the first place.

To achieve anonymity in pub-sub, it is crucial to avoid topologies with a central broker, because it would have full knowledge about publishers and their subscribers. To protect anonymity, a pub-sub system needs to distribute this brokering functionality onto all participants. While the distribution of brokering functionality prevents a single malicious participant from breaking anonymity, additional measures are required to prevent large-scale colluding eavesdroppers from inferring communication sources and sinks as publishers and subscribers from message flows.

* Corresponding author at: Technische Universität Darmstadt – CASED, Hochschulstr. 10, 64289 Darmstadt, Germany. Tel.: +49 61511623205.
E-mail addresses: daubert@tk.tu-darmstadt.de, kannx@kannx.net (J. Daubert), mfischer@icsi.berkeley.edu (M. Fischer), grube@tk.tu-darmstadt.de (T. Grube), stefan.schiffner@enisa.europa.eu (S. Schiffner), pkikiras@agtinternational.com (P. Kikiras), max@tk.informatik.tu-darmstadt.de (M. Mühlhäuser).

http://dx.doi.org/10.1016/j.comcom.2015.11.004
0140-3664/© 2015 Elsevier B.V. All rights reserved.

Nodes in such distributed systems take multiple roles simultaneously, namely the roles of publisher, subscriber, and broker. Information is disseminated from publishers, the root nodes in distribution trees, to subscribers in branch or leaf position. Additional nodes take over brokering functionality at branch positions and thus forward information on behalf of other nodes. The leakage of information of such distribution trees to attackers must be prevented for anonymous pub-sub.

The main contribution of this article is a distributed pub-sub system that is able to protect the confidentiality of the exchanged information as well as publisher and subscriber anonymity at the same time. In this context, we introduce two novel mechanisms to protect anonymity in distributed pub-sub: Probabilistic Forwarding (PF) and the Shell Game (SG) algorithm. PF protects the anonymity of subscribers in leaf positions of dissemination trees by letting them forward received information probabilistically to other non-subscribed nodes. SG protects the anonymity of subscribers by shuffling dissemination trees via position swaps of adjacent nodes. For the evaluation of these two mechanisms, we introduce the model of an anonymity attacker structured along three dimensions: scope, interaction, and knowledge of secret. We evaluate our distributed pub-sub system via extensive simulations by exposing it to a strong attacker with global scope, passive interaction, and without secret knowledge. Our simulations results indicate that we can achieve a protection of subscriber anonymity close to the optimum anonymity according to the presented metrics.

The rest of the article is structured as follows: Section 2 summarizes security and privacy requirements and introduces the model of the anonymity attacker. Section 3 discusses available technologies in the context of building blocks, followed by a discussion of existing systems. In Section 4, we present our existing anonymous and distributed pub-sub system and introduce two new mechanisms for protecting the anonymity of subscribers: PF and SG. We evaluated this system extensively via simulation by exposing it to the attacker and summarize our major evaluation findings in Section 5. Finally, Section 6 concludes the article.

2. Security and privacy in publish-subscribe

In this section, we introduce a notation and a model for distributed pub-sub systems. Furthermore, we discuss requirements for secure and privacy-preserving pub-sub and present an attacker model targeting the anonymity in such systems.

2.1. Formal pub-sub system model

A pub-sub system is established on top of a basic overlay that is a graph $G := (V, E)$ with peers V and edges $E \subset V \times V$. The set of attributes A represents user interests, e.g., the Twitter hashtag *#tahrir*. Per attribute $a \in A$, we have a set of subscribers $\mathcal{S}_a \subseteq V$ and a set of publishers $\mathcal{P}_a \subseteq V$.

On top of the basic overlay G and per attribute $a \in A$ an anonymous pub-sub system establishes an attribute mesh $\mathcal{M}_a := (V_a, E_a)$. $\mathcal{M}_a$ that connects all subscribers $\mathcal{S}_a$ with all publishers $\mathcal{P}_a$. To ensure connectivity of $\mathcal{M}_a$, P2P-based pub-sub systems, e.g., [3,4,6], distribute the brokering functionality among all nodes in the basic overlay. Nodes, which only take the broker role in an overlay $\mathcal{M}_a$ but not publisher and subscriber role, are called forwarders $\mathcal{F}_a$. The set of all nodes in $\mathcal{M}_a$ is denoted by $V_a := \mathcal{S}_a \cup \mathcal{P}_a \cup \mathcal{F}_a$.

Each node $v \in V$ has different kinds of neighbors. $N(v)$ as defined in Eq. (1) represents all neighboring nodes of v in the basic overlay G. The neighborhood $N(v)$ can be further differentiated within an attribute overlay. In inbound neighbors $N_a^+(v) \subseteq N(v)$ (Eq. (2)) forward notifications regarding a to the local node v; outbound neighbors $N_a^-(v)$ (Eq. (3)) receive notifications regarding a from node v.

$$N(v) = \{w\} : v, w \in V \wedge ((v, w) \in E \vee (w, v) \in E) \tag{1}$$

$$N_a^+(v) = \{w\} : v, w \in V_a \wedge (w, v) \in E_a \tag{2}$$

$$N_a^-(v) = \{w\} : v, w \in V_a \wedge (v, w) \in E_a \tag{3}$$

The messages exchanged in a pub-sub system are either of type advertisement, subscription, or notification. An advertisement m_a^{adv} for an attribute a is originated by a publisher $p \in \mathcal{P}_a$ and is disseminated by following the edges E of the basic overlay. A subscription m_a^{sub} is originated by a subscriber $s \in \mathcal{S}_a$ and establishes the set of overlay edges E_a. A notification m_a^{notif} is originated by a publisher p, and only traverses edges in E_a.

2.2. Requirements for privacy-preserving pub-sub

Manifold requirement definitions for secure and confidential pub-sub have been proposed, e.g., in [7,8]. We summarize these requirements and extend the anonymity definition .

2.2.1. Anonymity (privacy)

We define privacy in pub-sub as the combination of participant anonymity and confidentiality. Both requirements are closely related, since the lack of anonymity can lead to a violation of the confidentiality requirement and vice versa [9]. We use the set-based anonymity definition of Pfitzmann and Hansen: “Anonymity of a subject (participant) means that the subject is not identifiable within a set of subjects, the anonymity set.” [10]. In a communication system, the maximum anonymity set is the set of all participants, e.g., all nodes V. Even if an inside attacker compromises confidentiality, anonymity should be still preserved.

Subscriber anonymity ensures that every subscriber can hide in an anonymity set *anonSet* from the perspective of every attacker. An attacker attempts to establish the minimal anonymity set containing only subscribers, and thus disclosing these subscribers, for a given attribute a from a set of attribute $\mathcal{A}$. Thus, a pub-sub system meets subscriber anonymity w.r.t. to an adversary *adv*, and a set size k, if Eq. (4) holds. That is, the attacker cannot reduce any anonymity set below the size of at least k participants.

$$\forall a \in \mathcal{A} : |anonSet(a)| \geq k \tag{4}$$

Publisher anonymity ensures that a publisher cannot be identified within an anonymity set. Again, the size of the anonymity set provides a metric for the degree of the provided anonymity protection.

2.2.2. Confidentiality (privacy)

Information is kept and transmitted secretly between publishers and subscribers. That is, no outsider can access notification content (notification confidentiality), attributes (advertisement, subscription, and notification confidentiality), and communication metadata.

2.2.3. Scalability

A pub-sub system has to remain scalable in terms of number of supported nodes. For that, the overhead at single nodes should grow at most proportionally with the node number. Overhead incurs in terms of communication, computational, and memory overhead, and has to be minimal.

2.2.4. Authenticity and integrity

Only authorized participants can send notifications. Moreover, authenticity and integrity of messages can be verified.

2.3. Attacker model

An anonymity attacker on pub-sub systems can have different capabilities along the three dimensions: its activity level, its ability to infiltrate the system, and his attack scope. Regarding its activity level, an attacker can be completely *passive* and thus only eavesdrops messages. In contrast, an *active* attacker interacts with the system by modifying and spoofing messages. An *internal* attacker controls one or several malicious nodes within the system. Thus, he has knowledge about secrets, pseudonyms, and cryptographic keys, while the *external* attacker does not possess any secrets. The attacker is called *global* if it attacks on the basis of global network knowledge, and *local* or *colluding* if the attacker uses one or several colluding nodes for attacks.

The often referred *honest but curious* attacker [1,8,11] maps to the passive attacker with local scope and without secrets. We require an anonymous pub-sub system to protect against an active insider with local scope [6] (malicious node), as well as against a passive external attacker with global scope. This combined attacker is more powerful than the honest but curious attacker, and resembles NSA-like capabilities, i.e., large scale traffic monitoring in combination with compromised nodes. To extend our previous work, this article focuses on the passive external attacker with global scope.

2.3.1. *Global attacker*

The goal of this attacker is to break publisher and subscriber anonymity by monitoring all messages. One scenario for this attacker is intelligence organizations attempting to obtain network IDs of Online Social Network (OSN) users communicating about a certain attribute. For that, these organizations monitor the message flow of many Internet Service Providers (ISPs) to determine IP addresses of users using a certain attribute.

This attacker remains passive as it does not manipulate messages, external as it does not possess a priori key material, and global as it can observe the full communication network. According to the system model (cf. Section 2.1), the attacker observes notifications m^{notif}, knows an attribute a without associated secrets, possess topology information of the membership management $G := (V, E)$, and attempts to learn $\mathcal{S}_a$ or $\mathcal{P}_a$. The attacker can achieve this goal via statistical disclosure attacks [12–14]. With them, the attacker discloses message sinks and nodes that remain persistent in $\mathcal{M}_a$ after overlay restructurings as subscribers.

The attacker isolates $\mathcal{M}_a$, e.g., by observing notification flows. Assuming a very strong attacker that also knows the amount of publishers $|\mathcal{P}_a|$ and subscribers $|\mathcal{S}_a|$, the attacker sets up a probability distribution of all nodes V_a. In case publishers and subscribers dominate the overlay over forwarders, i.e., $|\mathcal{P}_a \cup \mathcal{S}_a| \div |V_a| > 0.5$, the attacker immediately breaks publisher and subscriber anonymity. In addition, the attacker uses timing information to isolate so called *leaf* nodes via message flow from $\mathcal{M}_a$ Such nodes have no outgoing neighbors in $\mathcal{M}_a$ ($N_a^-(v) = \{\}$). The attacker de-anonymizes such nodes as subscribers immediately in accordance with the overlay construction. We measure the attackers success via classification *accuracy* that is defined as the ratio of correct classifications (true positives TP and true negatives TN) among all classifications (TP, false positives FP, TN, and false negatives FN), Equation (5). Thus, the global attacker succeeds for *accuracy* > 0.5, i.e., it guesses publishers and subscribers better than a coin flip. In terms of anonymity size [10], *anonSet*(a) determines the set of nodes the attackers considers to be in $\mathcal{S}_a$ and $\mathcal{P}_a$.

$$accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{5}$$

3. Building blocks of anonymous publish-subscribe and related work

In this section, we discuss related work for anonymous pub-sub. The design of anonymous pub-sub leaves many choices for foundational technologies. We split anonymous pub-sub into six building blocks that enclose an elemental part of functionality each. In the following, we first discuss these building blocks and their underlying technologies. Second, we summarize related work along these building blocks.

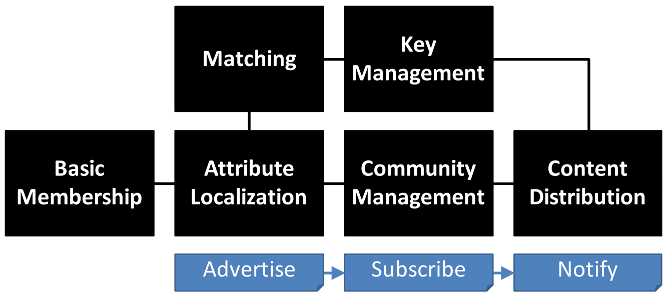

Fig. 1. Building blocks for anonymous pub-sub and their interdependencies (top). Correlation with pub-sub actions (bottom).

3.1. *Building blocks*

Fig. 1 provides an overview of the six building blocks and their interdependencies. The membership management interconnects nodes. The *attribute localization* facilitates the discovery of desired information and is supported by *matching* that provides privacy-protection on the basis of keys and cryptographic material derived during *key management*. Afterwards, mechanisms for *content distribution* ensure the delivery of the actual content. In the following we briefly discuss each of these building blocks.

3.1.1. *Membership management*

The membership management ensures the overall connectivity of the pub-sub overlay and it can be either centralized, decentralized, or distributed. In a centralized membership management, a single broker acts as intermediary (forwarder) between publisher and subscriber. Decentralized systems make use of interconnected brokers, so-called broker networks [15], and assign each node statically to one of these brokers. Centralized and decentralized systems have the drawback that one, or few nodes respectively, learn about all communication relations, which violates anonymity. In distributed membership management, each node in the system can take the role of a publisher, subscriber, and broker simultaneously. This requires each node to run a membership management protocol that usually follows a P2P approach. The employed P2P approach can be either unstructured, e.g., SCAMP [16], or structured, e.g., Scribe [17]. Structured protocols cause less signaling overhead, but the structure they impose on the resulting overlay can be exploited by an anonymity attacker.

3.1.2. *Attribute localization*

After a pub-sub overlay is established by the membership management, nodes need to localize attributes within it by querying brokers. A central broker that distributes attribute knowledge among publishers and subscribers, has global knowledge. In case such a broker gets compromised, the anonymity of the system is broken. Splitting this knowledge on multiple brokers helps, but does not solve the problem completely. When distributing brokering functionality onto all peers, a distributed localization service is required that either places attribute knowledge at rendezvous nodes or that floods the basic overlay with information on that attribute.

The usage of rendezvous nodes distributes attribute knowledge equally among all nodes. However, only one node remains responsible for each attribute and the attacker can attempt to monitor (global passive attacker) or compromise (passive insider) this node to break the anonymity for one attribute. Flooding eliminates this attack vector, but results in higher signaling overhead [18,19].

3.1.3. Key management

To ensure confidentiality and authenticity of exchanged messages, and thus to prevent the disclosure of any information that can break anonymity, key management is crucial for anonymous pub-sub. Keys and other cryptographic material, e.g., certificates, can be managed either by a central Trusted Third Party (TTP) or by a hierarchy of TTPs. When using a central TTP [20], anonymity can only be protected when the TTP and forwarders do not collude. The combination of keys and routing information can lead to the complete disclosure of participants and their interests. Furthermore, as a TTP is a Single Point of Failure (SPoF), a pub-sub system should not presume an online TTP. Hence, the TTP is only involved once per joining participant for the provisioning of the initial cryptographic material.

Rather than using a single TTP, keys can be also managed in a hierarchy, e.g., via a separation of attributes in groups and subgroups [21] a responsible TTP or TTP-role each.

3.1.4. Matching

After setting up the overlay and a localization service, notifications and advertisements have to be matched with subscriptions by forwarders. Matching might easily violate the confidentiality of publishers and subscribers, i.e., expose their interests. A multitude of techniques exist to implement the confidential matching between subscriptions and advertisements/notifications. The simplest form of matching is provided by using pseudonyms [22–24] or bloom filters [8,25–27]. Both can be compared efficiently, but Bloom filters additionally allow for an aggregation of attributes. Order-preserving cryptography [5,8,25,28] and Identity-Based Encryption (IBE) [4] can be also applied for attribute matching. They also enable a comparison of numerical values (greater than, less than, ...) and a text search (contains keyword). Zero-Knowledge Proofs (ZKPs) can be used to prove interest without revealing that interest [2], and seems to provide best protection of confidentiality, but are also computationally expensive and result in signaling overhead.

3.1.5. Community management

On top of the basic overlay provided by the membership management, nodes establish an additional mesh $\mathcal{M}_a$ per attribute a by making use of the attribute localization and matching service. These attribute meshes, or so-called communities, have to be maintained and optimized according to certain optimization goals in the presence of node churn. Such optimization goals can be the efficiency of the resulting mesh in distributing content, e.g., low delay between publishers and subscribers [29].

3.1.6. Content distribution

Once attribute meshes are established, the content distribution ensures the confidential delivery of authentic content while it needs to preserve the anonymity and low signaling overhead at the same time. The security of the content distribution strongly depends on the key management and on the membership management. Thus, it has to be considered in combination.

To distribute a notification, flooding $\mathcal{M}_a$ ensures delivery. With unreliable membership management, additional anonymous gossiping [30] can increase delivery reliability. However, modifications to anonymous gossiping are required. For instance, knowledge of the distance to the next publisher and subscriber should be avoided to protect anonymity.

Regarding anonymity and confidentiality, notifications in content distribution should not reveal more information than advertisement in the attribute knowledge building block. For that, the content of notifications must be encrypted. While symmetric cryptography is fast, multilayer encryption such as onion routing [31], Multiple Layer Commutative Encryption (MLCE) [32] must be considered with care due to their computational overhead.

3.2. Related work in anonymous pub-sub

Anonymity as a requirement in anonymous pub-sub has been first introduced by Wang et al. [7]. In particular, they point out that pub-sub may provide anonymity against the passive internal attacker, assuming each path between publisher and subscriber contains at least three nodes (cf. onion routing [33]) in the basic overlay. However, anonymity protection remains in a theoretical stage. Furthermore, challenges of key management are not resolved as pairwise keys are required and key management is left as open.

Srivatsa and Lui [23] as well as Raiciu and Rosenblum [8] address key management and matching via an online TTP and pseudonyms. However, they require a central broker, or broker network respectively, and thus cannot provide publisher and subscriber anonymity. Chen et al. [5] distribute the broker role, but leave key management out-of-band. Their approach uses a searchable ciphertext method for matching.

At the same time, first Peer-to-Peer (P2P)-based approaches that address key management as well as anonymity appeared, e.g., Shikfa et al. [3,32] and Tariq et al. [4]. However, the overlay networks created by these approaches still leak information that can be exploited by an attacker to break anonymity: with the approach of Shikfa et al., public keys of several nodes leading towards the recipient are required to be known at all times (passive internal attacker, passive global attacker). Hence, the attacker can reason about the overlay position of the recipient. With the approach of Tariq et al., subscribers learn about other subscribers with similar attributes (passive internal attacker).

Following these P2P solutions, further advancements have been made in the area of key management and matching, such as the work from Nabeel et al. [1,2] (homomorphic cryptography) and Di Crescenzo et al. [11] (oblivious transfer (OT)). These mechanisms protect anonymity and confidentiality against the internal attacker. However, message flows can be still attacked by the global passive attacker to break anonymity.

Summarizing the related work, none of them provides anonymity to both, publisher and subscriber, against both, the global and the internal attacker, while still being scalable. Several systems, which provide confidentiality, depend on expensive key exchange mechanisms. Thus, scalability and minimal overhead requirements are not fulfilled. None of those systems achieves all of the listed requirements at once.

4. AnonPubSub

In this section, we describe our anonymous pub-sub system that builds up on a system formerly introduced by us in [6]. We significantly enhance this work by providing additional resistance against an anonymity attacker that has global knowledge on all communication in the system. The former system provided anonymity protection against an internal attacker with a very restricted view on the overall system. To protect anonymity in the presence of the more powerful global attacker, we introduce Probabilistic Forwarding (PF) and the Shell Game (SG) as two novel anonymity-preserving mechanisms for pub-sub.

Fig. 2 shows an example for our system. We presume a basic overlay network that is established by a membership management approach. A key management ensures the exchange of cryptographic material. On that basis, publishers (squares 1, 5) distribute attribute knowledge via a basic overlay, so that subscribers (rhombuses 7, 9) can match their interests. Afterwards, they establish an attribute overlay by sending anonymity-preserving subscriptions via the basic overlay to the publishers. As a result, all nodes on the subscription path will be included as brokers to the attribute overlay and thus will potentially also include non-subscribers or forwarders (circles 3, 6)

The system ensures that the TTP is not involved in the actual message dissemination. Forwarders ensure a connected attribute overlay

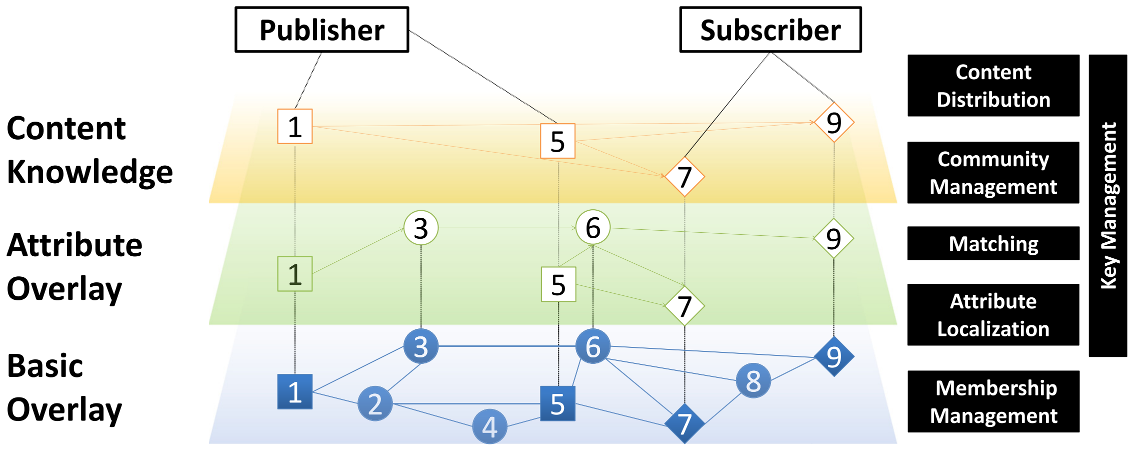

Fig. 2. Membership management (bottom), pub-sub overlay (middle), community (top). Square: publisher, rhombus: subscriber, circle: forwarder.

without requiring publishers and subscribers to know each other, and thus anonymity is protected.

To describe our system, the remainder of this section follows the structure of the building blocks on the right side of Fig. 2. Within community management, we introduce SG as novel approach for anonymity protection. Within content distribution, we provide a novel adaptation of cover traffic (PF) to overlay networks, to protect anonymity as well.

4.1. Membership management

IP serves as communication protocol between peers for membership management. The gossiping protocol SCAMP [16] maintains the neighborhood, i.e., the set $N(v)$ (cf. Function (1)). Successive building blocks use UDP over IP to ensure every packet is well controlled and no undesired information leak occurs. Heartbeat messages between neighbors support the detection of connection failures. Furthermore, a padding of these heartbeats creates a covert channel for successive building blocks. Heartbeats carry messages of the other building blocks. The padding prevents the global passive attacker from inferring message type from size. As an exception, we allow content distribution to send notifications outside of heartbeats. As a result, membership management does not restrict content distribution in terms of delivery time, size, or rate.

4.2. Key management

An offline TTP performs key management with access control. The TTP serves as trust anchor by providing joining nodes with keys to subscribe and to decrypt attribute content. Furthermore, it issues certificates for publishers, so that they can disseminate authenticated content. The TTP has a secret private key sk_{TTP} and a public key pk_{TTP} that is known by all participants.

Whenever a new publisher joins the system, he creates a secret/public key pair (sk_a, pk_a), and obtains a certificate $cert_a$ on his public key pk_a from the TTP. Afterwards, the publisher can sign advertisements and notifications with sk_a and attaches $cert_a$. Hence, every receiving node can verify the authenticity of received advertisement and notification messages. In addition, the TTP hands out a key K_a to all publishers P_a and all subscribers S_a of attribute a. This key is used for protecting the confidentiality of exchanged messages for attribute localization, matching, and in content distribution. Hence, only legitimate nodes can subscribe to an attribute a and access the respective content.

In addition to the keys provided by the TTP, neighboring nodes in the established overlay exchange pair-wise keys to secure all exchanged messages. As a result, every message looks different on each hop, thus impeding the analysis of the system from the outside.

4.3. Attribute localization

To localize attributes, knowledge on them needs to be distributed in the system. For that, we flood the basic overlay G per attribute a with advertisement messages. Every advertisement contains an attribute pseudonym for matching as well as a novel transaction pseudonym to prevent basic overlay routing loops and to optimize the resulting attribute overlay.

To construct an advertisement message, a publisher $p \in \mathcal{P}_a$ derives an attribute pseudonym t_a from attribute a by encrypting it with the shared secret K_a, so that $t_a = \{a\}_{K_a}$. In addition to t_a, the publisher creates a random transaction pseudonym, a nonce h, to distinguish multiple publishers for the same attribute and thus prevents overlay partitioning and loops. The publisher then attaches t_a, h to the advertisement message m_{adv}, and sends the message to all its neighbors $N(p)$ in G. The transaction pseudonym allows nodes in G to resolve the challenges of duplicate detection and path length minimization as illustrated in Fig. 3. Left: without global node identifiers, f_1 could not detect the loop without h. Right: f_4 cannot detect duplicates without h.

In addition to loop and duplicate detection, we include a novel mechanism for overlay path optimization. It allows to use the shortest paths in the basic overlay between publishers and subscribers by avoiding a hop count field that could be exploited in an attack. Nodes do not simply forward an advertisement message unmodified, but instead hash the included nonce h via a cryptographic hash function H to value $h' = H(h)$ first. As a result, instead of a single transaction pseudonym, we are using a hash chain per path. Upon reception of the same advertisement messages via two different neighbors, it enables a node to compare the length of the two possible subscriptions paths. For that, it needs to check if one hash value can be mapped on the other via successive hashing. The value that can be mapped onto another, has its origin from a shorter path in the basic overlay.

Using a hash chain rather than transaction pseudonym and hop counter yields two major benefits: First, a hash chain does not reveal the distance to the publisher (publisher anonymity). Second, active internal attackers cannot lie (decrement) about the hash chain to gather more traffic.

The publisher initializes a hash chain with nonce h and attaches it together with t_a to an advertisement message $m_{adv} = (t_a, h)$. Forwarders keep the triples (t_a, h, v) in routing tables, where v is the neighbor it received the message from. When a forwarder receives an already known t_a, it checks if one value can be mapped onto the other to get to know if both transaction pseudonyms originate from the same publisher and if there is a new and shorter path.

Fig. 4 provides an example: $p \in \mathcal{P}_a$ obtains $cert_a$ from the TTP and creates an advertisement message $m_{adv} = (t_a, t_{a\,sig}^{sk_a}, cert_a, h)$ contain-

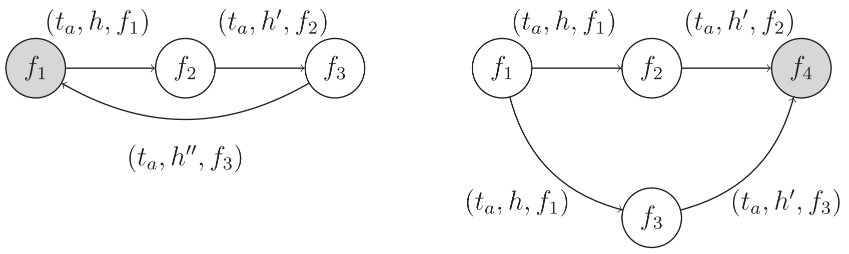

Fig. 3. Use of transactions pseudonyms. Left: loop detection. Right: duplicate detection.

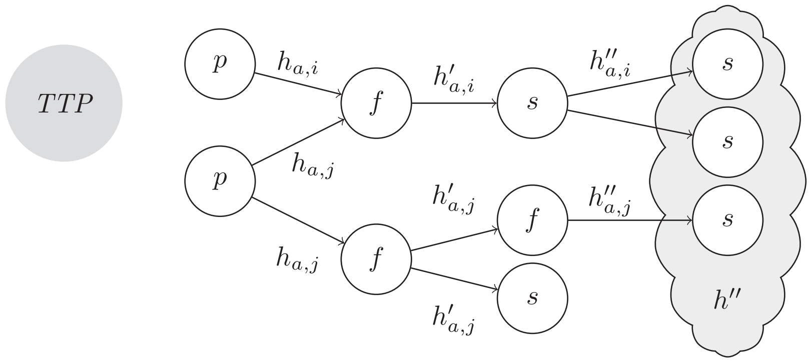

Fig. 4. Distribution of advertisements. Forwarder increment hash chain elements ($h \rightarrow h' \rightarrow h''$) and merge advertisements when forwarding.

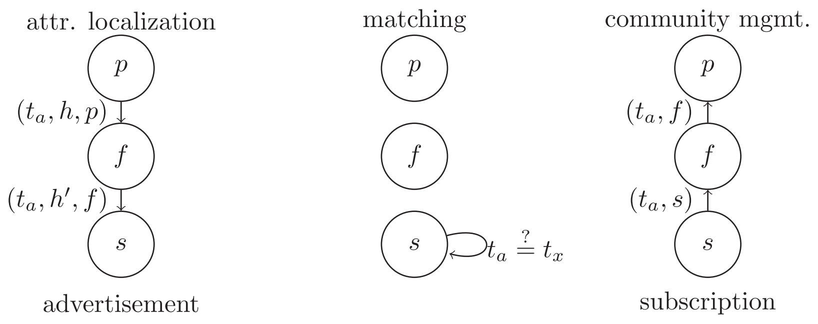

Fig. 5. Process of attribute localization, matching, and community management.

ing the encrypted attribute, the hash value, the signature, and the certificate. Afterwards, it floods the advertisement to all its neighbors $N(p)$.

4.4. Matching

Nodes perform matching via binary comparison of pseudonyms. For that, subscribers prepare attribute pseudonyms t_a for their attributes a, and compare the pseudonym with incoming advertisements. In case of a match, a subscription is send on the reverse path of the respective advertisement.

Fig. 5 illustrates the process of attribute localization (left) via advertisements, matching (center), and community management (right) via subscriptions. Here, t_a is the attribute pseudonym of the advertisement, and t_x the interest of subscriber s. When the system contains multiple publishers for a, every publisher $p \in P_a$ that receives an advertisement for a subscribes to it. This ensures the connectivity of overlay $\mathcal{M}_a$.

4.5. Community management

Attribute overlays, so-called communities, must be maintained under churn. Furthermore, resulting maintenance operations must ensure anonymity and confidentiality at the same time.

We use heartbeat messages to detect node and connection failure, as well as means to exchange management messages in a privacy-preserving manner. For that, we increase the packet size of heartbeat messages via padding to enclose the largest possible management message. Now management messages can be “piggy-backed” inside heartbeats.

Nodes leaving the attribute overlay send unadvertise and unsubscribe messages to their neighbors. Likewise, a node detecting a neighbor’s failure acts like having received such a message from the neighbor. This ensures that obsolete routing table entries are purged and replaced with alternative entries.

The *Shell Game (SG)* shuffles the overlay topology to prevent the global passive attacker from inferring roles from the topology. We use a method inspired by the real-life *shell game* to restructure an overlay network, without that a global attacker can learn its actual structure. In a shell game, a ball is hidden under a shell. Then the positions of the shells are swapped, so that a player cannot tell the shell with the ball from empty shells.

In our pub-sub SG, nodes swap overlay positions by exchanging their neighborhood sets. We hide those exchanged messages within heartbeat messages (cf. community management) and thus prevent the attacker from observing position changes. Thus, we call this approach *SG algorithm*.

Every node starts the SG as soon as it has at least one overlay neighbor. When an overlay $\mathcal{M}_a$ is created by connecting subscribers with publishers along their subscription paths, $\mathcal{M}_a$ is immediately shuffled even before the first notification reaches the subscribers. Hence, the global attacker cannot observe $\mathcal{M}_a$ “before”

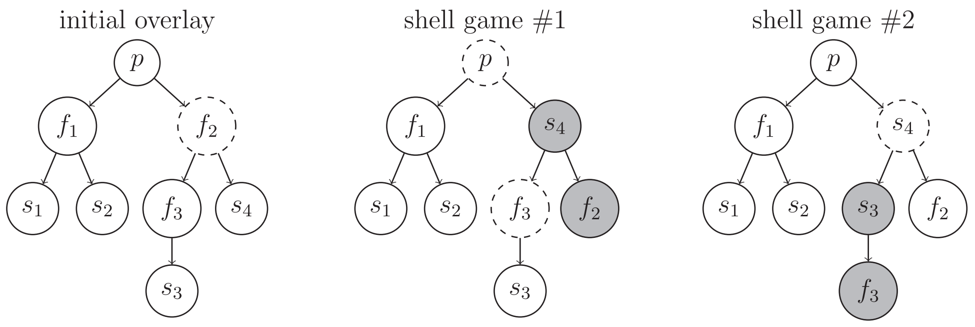

Fig. 6. Initial state and two steps of the SG. The active node is marked with a dashed outline, swapped nodes are marked in gray.

the SG takes place. As a result, the subscriber anonymity set for the global attacker equals the size of the whole overlay: $|anonSet(a)| = |V_a|$. Furthermore, the attacker accuracy for classifying subscribers remains as low as random guessing $accuracy = |\mathcal{S}_a| \div |V_a|$.

A node v first executes the Procedure *shellGame* upon the connection of the first subscriber. An exponential decay function (line 1) terminates the SG. The attribute overlay membership time $t(v)$ of the node constitutes the variable parameter. Now, the probability of v performing a SG decreases over time (lines 2–3). Nodes not playing the SG schedules another invocation after an exponentially distributed waiting time (lines 4–7). Otherwise, v chooses a neighbor $w \in N_a^-(v)$ in (line 8, cf. Function (3)), and sends a message to w asking to swap positions in the overlay (line 9). As a result, w hands over $(N_a^-(w), N_a^+(w))$ to v, and notifies these neighbors. Then v completes the SG by handing over $(N_a^-(v), N_a^+(v))$ to w and notifies these neighbors as well (lines 10–13).

Procedure shellGame(N_a^+(v), N_a^-(v), t(v)).

```
$P_{swap} \leftarrow e^{-\lambda * t(v)}$;
r ← randomUniform(0,1);
if $r > P_{swap}$ then
    $t_w$ ← randomExponential();
    wait($t_w$);
    shellGame($N_a^+$(v), $N_a^-$(v), t(v)+$t_w$);
    return;
swapee ← pickUniform($N_a^-$(v));
$N_a^-$(v) ← message(swapee, new children: $N_a^-$(v) \ swapee);
for $n \in N_a^-(v) \setminus swapee$ do
    message(n, new parent: swapee);
for $n \in N_a^+(v)$ do
    message(n, swap child: n by swapee);
return;
```

Fig. 6 depicts an overlay with one publisher, four subscribers, and three forwarders (left). The SG is performed two times, the outcomes are shown at the center/right of Fig. 6. We mark overlay changes between subsequent SGs in gray and highlight the node that executes the algorithm as dashed.

In Fig. 6, subscriber s_4 initially joins the overlay, and its parent node f runs the algorithm to swap with s_4 as depicted in step (center). After that, the nodes p and f_3 obtain new neighbors and also run the algorithm. In step 2, f_3 swaps with child s_3 while p does not swap. Then, s_4 detects another change, new neighbor s_3, and executes the SG algorithm.

Parallel SGs may affect intersecting sets of nodes and could result in a partitioned overlay. The two-phase commit protocol [34,35] resolves this issue by locking adjacent nodes before starting the SG. We resolved conflicting locks via a timeout that has the same effect as waiting for t_w. The protocol also ensures that no notifications are lost. Nodes locked via the two-phase commit do not delete routing table entries until the SG is complete and the lock is removed. As a result, duplicate notifications may occur within the set of locked nodes.

4.6. Content distribution

Content distribution handles the dissemination of notifications. Symmetric cryptography ensures confidentiality; digital signatures ensure integrity and authenticity.

Publishers distribute notifications for attribute a by flooding the notification through the respective and pre-established attribute overlay $\mathcal{M}_a$. With attribute pseudonym, encrypted content, and signature a notifications becomes $m_{notif} = (t_a, \{m\}_{K_a}, sig)$ with $sig = sign(t_a||\{m\}_{K_a}, sk_a)$. The notation $x||y$ denotes the concatenation of elements x, y.

The global passive attacker applies the methods introduced in Section 2.3 to de-anonymize subscribers. That is, exploit the role distribution of the attribute overlay, as well as to pick leaf nodes. For the role distribution, the attacker has an advantage picking subscriber from an overlay mainly constituted by subscribers. Likewise, before the SG is played, the attacker has an advantage picking leaf nodes as subscribers.

To mitigate this attack, we adapt the concept of cover traffic [36] as *Probabilistic Forwarding (PF)*. PF adds nodes, which are not part of the attribute overlay $\mathcal{M}_a$, into the overlay. As a result, the number of nodes in $\mathcal{M}_a$ increases. Moreover, the global attacker loses the advantage of picking leaf nodes as subscribers. Compared to related work [36], we add cover nodes to the attribute overlay rather than just relaying messages to them. As a result, these nodes become indistinguishable from normal nodes. This is possible as our overlay construction supports the notion of forwarders. Compared to the SG, PF can be applied faster as the protocol overhead is lower.

PF works as follows: upon sending a subscription message, an overlay node $v \in V_a$ decides for every basic overlay neighbor $w \in N \setminus V_a$ at random to forward notifications to this neighbor. That is, to add w to $N_a^-(v)$ (cf. Function (3)). Node w also becomes member of V_a and $\mathcal{F}_a$. The system parameter $\mu \in [0, 1]$ models the probability of choosing w. This concept causes the overlay to grow, and thus $anonSet(a)$ ultimately increases as well.

Fig. 7 exemplifies this concept. Left: subscribers s_1, s_2, s_4 are exposed as leaf nodes. The attack accuracy evaluates to $(3+1) \div (3+0+1+1) = 0.8$. Middle: subscribers s_1, s_3 conceal themselves by adding cover neighbors. Thus, the accuracy drops to ≈ 0.43. Right: cover neighbors may pick additional cover neighbors by themselves—cover node f_3 pulls in another cover node, f_4 (accuracy 0.5).

PF causes signaling overhead during content distribution. Furthermore, PF does not change the inner topology of the overlay. Hence, the global passive attacker can still infer information from this structure. To compensate the latter drawback, PF can be combined with SG as Probabilistic Forwarding & Shell Game (PFSG).

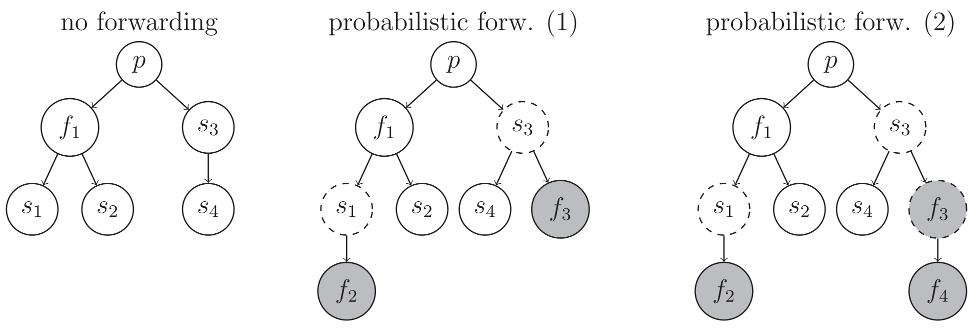

Fig. 7. Hiding nodes dashed, receivers of cover messages gray. Middle: PF with one hop of neighbors. Right: two hops of neighbors.

4.7. *Security discussion*

The presented system seems to meet all requirements to anonymous pub-sub systems from Section 2.2.

4.7.1. *Anonymity against the internal attacker*

AnonPubSub ensures subscriber anonymity against the non-colluding internal attacker with anonymity sets $anonSet \geq |N_a^-(v)| + 1$.

For publisher anonymity, we assume an internal attacker that possesses key material and imposes a subscriber. The attacker constructs a tree in G, rooted by himself that contains the shortest paths to all nodes. The attacker then removes branches from which he does not receive advertisements on the shortest path. Thus, the remaining branches contain at least one publisher each, and the size of each branch forms the anonymity set *anonSet* for the contained publisher. The publisher can estimate *anonSet* to be at least of size $|N_a^+(v)| + 1$— his predecessors in the mesh plus himself. The same argument holds for subscriber anonymity assuming an internal attacker imposing a publisher.

AnonPubSub is secure against this attacker, even without PF and SG. With PF, the set $\mathcal{F}_a$ increases. Anonymity still holds at the behavior of $\mathcal{F}_a$ remains indistinguishable from other participants. Likewise anonymity holds for SG as all nodes in V_a equally play SG and thus remain indistinguishable to the internal attacker as well. The avoidance of node identifiers and hop counts ensures that an active internal attacker does not learn the roles of neighboring nodes while playing the SG or being involved in PF. Section 5 complements this discussion with an analysis of anonymity against the global passive attacker.

4.7.2. *Confidentiality*

The system provides *notification confidentiality* and *subscription confidentiality*. In particular, attributes are only transmitted via pseudonyms. Notifications are only transmitted encrypted. The TTP ensures only members of closed attribute groups obtain the key material.

4.7.3. *Scalability*

In terms of memory overhead, AnonPubSub scales proportionally with the number of attributes and neighbors. As for the communication overhead, attribute knowledge cause the most signaling costs due to flooding of the basic overlay. The costs for subscriptions and notifications are bounded by $diameter(G) \times |\mathcal{S}_a|$ where $diameter(G)$ denotes the diameter of the basic overlay.

PF and SG cause additional signaling costs. Each SG between two nodes v_1, v_2 requires $3 \times |N_a^+(v_1)| + |N_a^+(v_2)| + 3 \times |N_a^-(v_1)| + |N_a^-(v_2)|$ messages including the two-phase-commit protocol. We can approximate $|N_a^+(v_x)| \approx 1$ and therefore require $3 \times |N_a^-(v_1)| + |N_a^-(v_2)| + 4$ messages. The actual signaling costs of the SG as well as PF highly depend on the basic overlay topology and have to be analyzed further via simulation in Section 5.

4.7.4. *Integrity and authenticity*

The system enforces integrity and authenticity via signatures of advertisement and notification messages to prevent local attackers from spamming attributes and notifications. Subscriptions do not require authenticity as they do not threaten scalability.

5. Evaluation

After exposing the system to the internal attacker in the previous section, we now analyze AnonPubSub with respect to the global attacker (cf. Section 2.3). For that, we developed a simulation framework and conducted a study.

With this study, we answer the following research questions:

- What are the influences of node degree ($|N(v)|$) and the ratio of subscribers on the attackers classification accuracy?
- Which forwarding probability μ provides best trade-off between low attack accuracy and low signaling costs?
- Which SG parameter λ provides best trade-off between low attack accuracy and low signaling costs?
- How should parameters (μ, λ) for the combination PFSG be chosen to achieve an equilibrium of forwarders and subscribers?

5.1. *Simulation setup*

To evaluate our system, we implemented a simulation model for the OMNeT++[1] discrete event simulator and by making use of the INET framework to connect nodes via IPv4.

We assume a fixed basic overlay as determined by the membership management for our experiments. We simulate one attribute overlay with one publisher as a strong global attacker can eliminate the occlusion of multiple overlays and publishers. We use small-world networks generated according to the Watts–Strogatz model [37], initialize each node with its neighborhood set $N(v)$, and simulate with parameters and metrics as summarized in Table 1.

The parameter μ models the probability of each node for SG. The parameter λ controls the exponential decay of SG. We simulate four variations of AnonPubSub with this setup: Baseline (BL) for the system without extensions, Probabilistic Forwarding (PF), Shell Game (SG), and Probabilistic Forwarding & Shell Game (PFSG) for the combination of all protective measures. We measure the number of overlay nodes in comparison to all membership management nodes, the signaling costs in messages, the number of performed SGs, the ratio of leaf subscribers among all leaf nodes within the overlay, and the global attackers accuracy for classifying subscribers.

5.2. *Influence of basic overlay properties on the attack accuracy*

To analyze the influence of the basic overlay, we use $|S_a|/|V| = 0.1$ subscribers and vary $|N(v)| \in [2, 16]$. For PF, we set $\mu = 0.7$. For SG,

[1] http://www.omnetpp.org/ .

Table 1
Simulation parameters and metrics.

Parameters	
$\lvert V\rvert = 500$	Size of the basic overlay
$\lvert S_a\rvert/\lvert V\rvert \in [0,1]$	Ratio of subscribers among $\lvert V\rvert$
$\lvert S_a\rvert/\lvert V_a\rvert \in [0,1]$	Ratio of subscribers among $\lvert V_a\rvert$
$\lvert\mathcal{P}_a\rvert = 1$	Number of publishers per attribute
$\lvert\mathcal{A}\rvert = 1$	Number of attributes in the system
$N(v) \in [2,16]$	Node degree in the basic overlay
$runs = 30$	Repetitions per setup
$\mu \in [0,1]$	PF probability
$\lambda \in [0,1]$	SG decay
Metrics	
$\lvert V_a\rvert \in [1, \lvert V\rvert]$	Size of the attribute mesh
$costs \in \mathbb{N}$	Signaling costs in #messages
$swaps \in \mathbb{N}$	Performed SGs
$\lvert\mathcal{S}_a \cap L(\mathcal{M}_a)\rvert/\lvert L(\mathcal{M}_a)\rvert \in [0,1]$	Ratio of leaf subscribers in $\mathcal{M}_a$
$accuracy \in [0,1]$	Subscriber classification

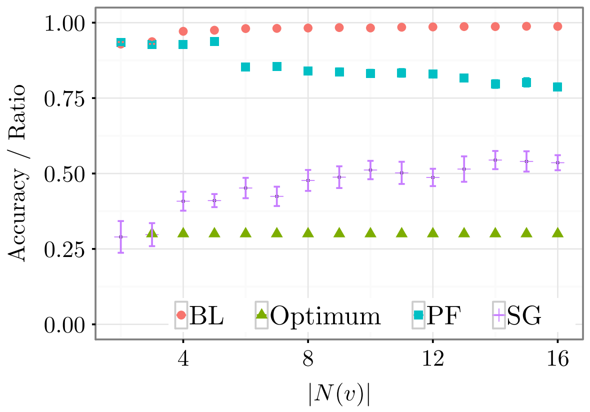

Fig. 8. Attack accuracy over $|N(v)|$ for BL, PF, and SG compared to anonymity-optimal accuracy.

we set $\lambda = 0$ (no decay) and measure all metrics after 500 SGs. The simulations indicate that the subscriber ratio of 0.1 creates overlay networks that span across the whole diameter of the basic overlay. Furthermore, $\mu = 0.7$ seems to balance anonymity and signaling overhead.

For *BL*, we expect the classification accuracy to increase with the node degree. A higher node degree leads to shorter overlay paths and thus fewer forwarders. PF and SG should both benefit from a higher node degree and lead to a lower accuracy.

Fig. 8 shows the accuracy for the three experiments compared to anonymity-optimal accuracy (chances of randomly picking a subscriber). With only the BL system, the attacker picks subscribers with high accuracy (0.927 for $|N(v)| = 2$, 0.988 for $|N(v)| = 16$). PF only provides protection with at least 4 neighbors and benefits from higher node degree (accuracy of 0.787 for $|N(v)| = 16$). The SG provides the best anonymity protection (accuracy of 0.29 for $|N(v)| = 2$). However, the protection degrades with higher node degree due to the increasing lack of forwarders. As reference, $|S_a|/|V_a|$= 0.299 determines the chances of randomly picking subscribers from overlay nodes (optimum).

Next, we vary $|\mathcal{S}_a|/|V|$ to evaluate the influence of the ratio of subscribers on the accuracy of the global attacker. We set $|N(v)| = 4$, simulate with $|\mathcal{S}_a|/|V| \in [0.1, 0.9]$, and measure the global attacker's classification accuracy for PF and SG (after 500 SGs).

With an increasing ratio, we expect more subscribers to take the broker role, and thus become harder to detect. Hence, the classification accuracy for PF should drop. However, high ratios limit the availability of nodes for PF. As a result, the accuracy should stabilize or increase for high ratios. For SG, low ratios should lead to the

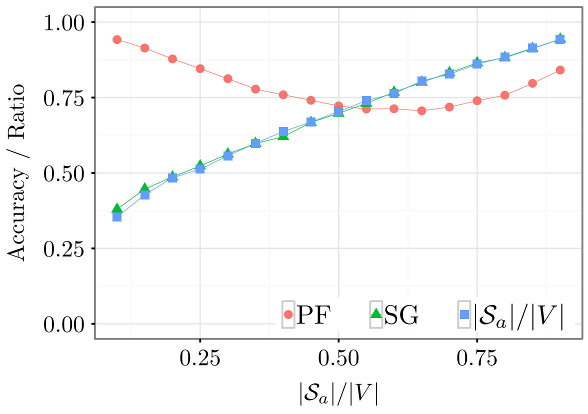

Fig. 9. PF/SG. Attack accuracy in dependence on $|S_a|/|V|$ for PF and SG.

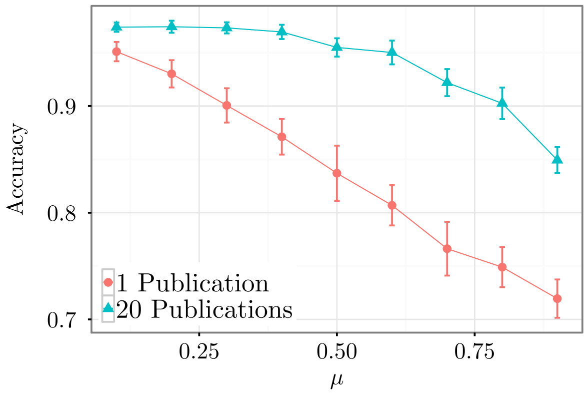

Fig. 10. Attack accuracy over μ, each after $publications = \{1, 20\}$.

lowest attack accuracy as many forwarders in the overlay can be used to swap.

Fig. 9 shows the attack accuracy compared to the subscriber ratio. The accuracy for PF reaches its lowest point with $|S_a|/|V| = 0.65$. Here, $\mu = 0.7$ leads to the optimal mixture of subscribers and publishers within the overlay. As for the previous experiment, the SG provides optimal anonymity protection.

5.3. Influence of the forwarding probability on the attack accuracy

To analyze the influence of μ as well as neighbor selection, we simulate PF with $\mu \in [0.1, 0.9]$ and fix $|\mathcal{S}_a|/|V| = 0.1$ again. We expect a proportional decreasing attacker accuracy when increasing μ, as this corresponds to a ratio of μ leaf nodes covering themselves with PF. Likewise, we expect proportional growth of signaling overhead.

Fig. 10 shows the decreasing attack accuracy over increasing μ. With a fixed neighbor selection (observation of 1 publication), the accuracy drops proportionally as expected. However, with a variable neighbor selection (observation over 20 publications), the accuracy increases significantly. To analyze this effect, we simulate for $\mu = 0.7$ and $|\mathcal{S}_a|/|V| = 0.3$, and measure after each publication. Fig. 13 shows an increase in attack accuracy with every publication. Therefore, PF should select neighbors only once.

Fig. 11 shows the signaling costs of PF. With 20 publications, it becomes evident that PF causes overhead over time for high μ. Furthermore, PF does not reach the anonymity optimum ($accuracy = 0.5$). Still, PF does not reach worst case costs of $479 * 20 = 9580$ messages for 20 publications as an anonymity optimal broadcast scheme would do. Fig. 12 explains this increase in signaling costs according to the overlay size. Compared to the BL with an attribute mesh of 141 nodes, PF with $\mu = 0.6$ already doubles the overlay size (282 nodes).

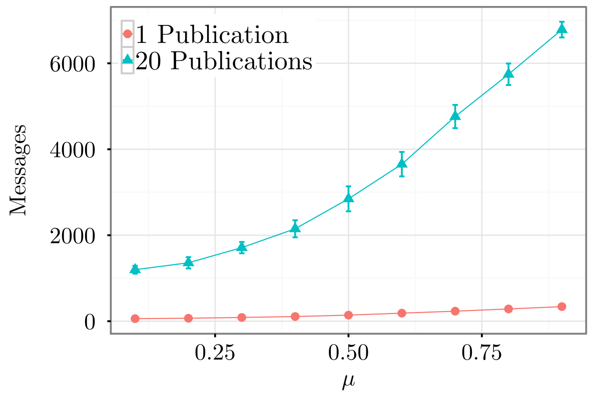

Fig. 11. Sent messages over μ, each after *publications* = {1, 20}.

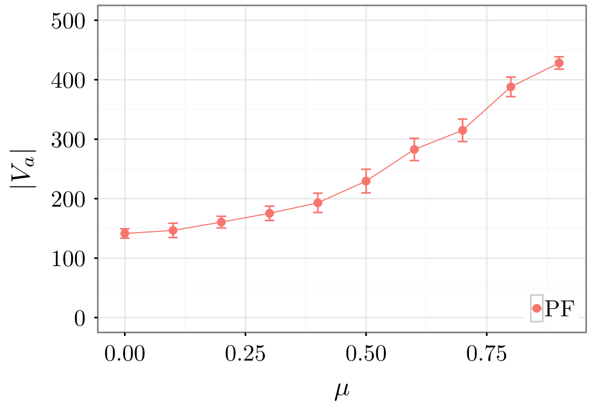

Fig. 12. PF. Overlay size in nodes in dependence on μ for PF.

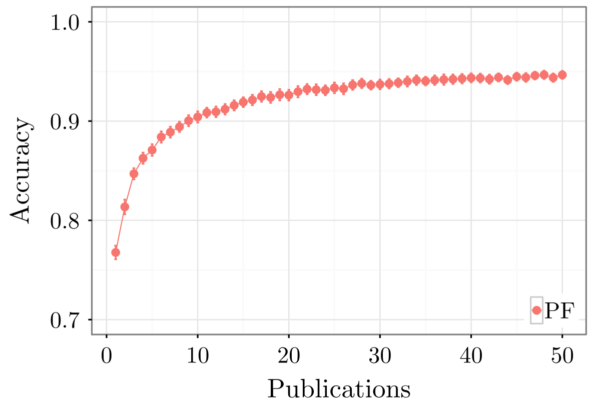

Fig. 13. PF. Attack accuracy in dependence on *publications* for PF.

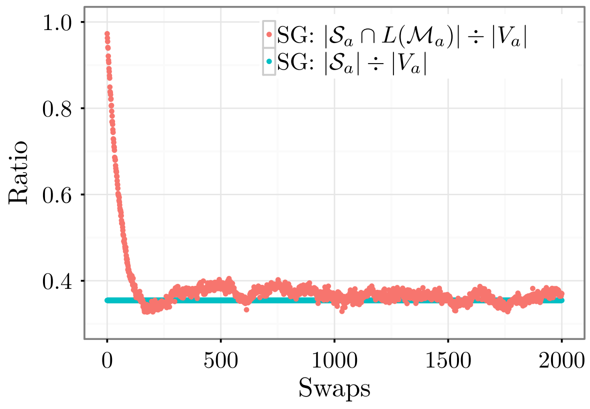

Fig. 14. Leaf subscriber ratio over *#swaps* compared to the optimal leaf subscriber ratio for anonymity. Time decay.

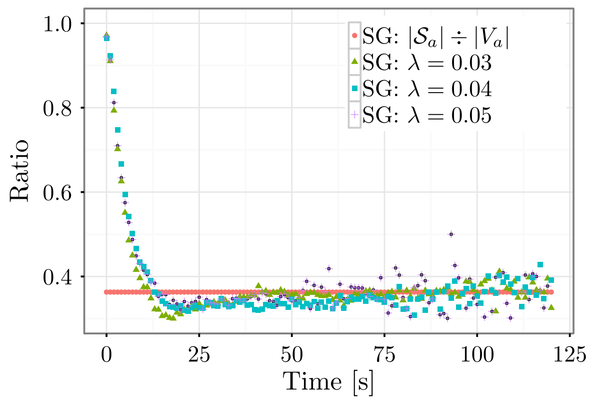

Fig. 15. Leaf subscriber ratio over time for $\lambda = \{0.03, 0.04, 0.05\}$ compared optimum ratio. Time decay.

5.4. Influence of the SG parameter on the attack accuracy

The exponential decay function (cf. Section 4.5, Procedure shellGame) controls the behavior of SG. To determine the optimal value for parameter λ (optimal anonymity with minimal signaling costs), we vary $\lambda \in [0, 0.05]$, and measure the ratio subscribers among leaf nodes ($|\mathcal{S}_a \cap L(\mathcal{M}_a)|/|L(\mathcal{M}_a)|$). The ratio of subscribers among leaf nodes tells us the anonymity optimal mixture. The number of SGs serves as signaling cost indicator. To prevent colliding SGs of adjacent nodes, nodes initiate the SG at random times drawn from the exponential distribution with mean 5 s.

We expect that every node must swap once across the diameter of the overlay mesh, i.e., the height of the tree rooted by the publisher. With $|N| = 500$ and $|N(v)| = 4$, we estimate this tree height within $log_4(500) = 4.48$. For the number of SGs, we assume $141 \times 4.48 =$ 587 ($|V_a| = 141$) SGs to be sufficient to shuffle the whole overlay. As nodes play SGs independently, we expect a duration of 4.48×5 s $= 22.4$ s until the system reaches the optimal leaf node mixture.

Fig. 14 shows the average convergence of the leaf subscriber ratio towards the overlay subscriber ratio without decay ($\lambda = 0$). The ratio converges indeed at 587 SGs. However, the SG also reaches optimal ratio once before this convergence. This occurs as subscribers already play the SG before the attribute mesh is connected. Subscribers then tend to swap into leaf positions as the diameter of overlay partitions is low in this stage. Hence, the ratio of leaf subscriber fluctuates before reaching convergence.

Fig. 15 shows the convergence for $\lambda \in \{0.03, 0.04, 0.05\}$ over time. The ratio $|\mathcal{S}_a \cap L(\mathcal{M}_a)|/|L(\mathcal{M}_a)|$ reaches the target ratio before the expected 22.4 s, as for the number of SGs. However, $|\mathcal{S}_a \cap L(\mathcal{M}_a)|/|L(\mathcal{M}_a)|$ converges slower towards the optimal ratio than expected. This is due to collisions of SGs played by adjacent nodes, which cause delay for some SGs. Hence, it takes longer than 22.4 s to swap 587 times (≈ 4.22 SGs initiated by every mesh node). Furthermore, we can see that higher values of λ, e.g. 0.05, cause scatter over time. This happens as the decay function reduces the rate of SGs. Thus, successive plot points are based on fewer measurements. We chose $\lambda = 0.05$ as scatter starts once anonymity optimal accuracy is reached.

Fig. 16 illustrates why the decay function depends on the overlay membership time rather than the number of SGs. By eliminating the influence of the decay function with $\lambda = 0$, the SG works as expected (bottom plot). However, with a fast decay ($\lambda = 0.5$) the SG converges away from optimum. The same effect occurs for smaller values as used in Fig. 16. That happens, as nodes close to leaf

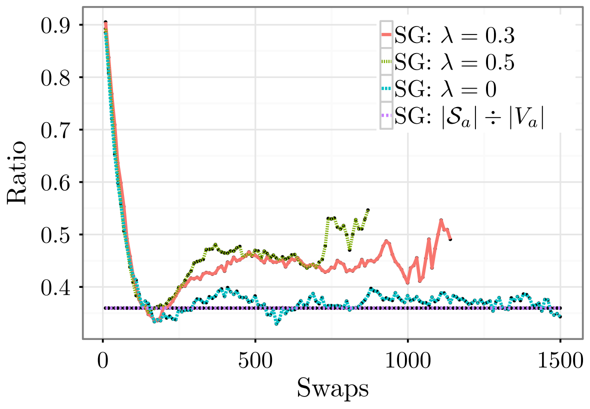

Fig. 16. Ratio of leaf subscribers over *#swaps*. Each for $\lambda = \{0, 0.3.0.5\}$ compared to the optimal leaf subscriber ratio for anonymity. Swap decay.

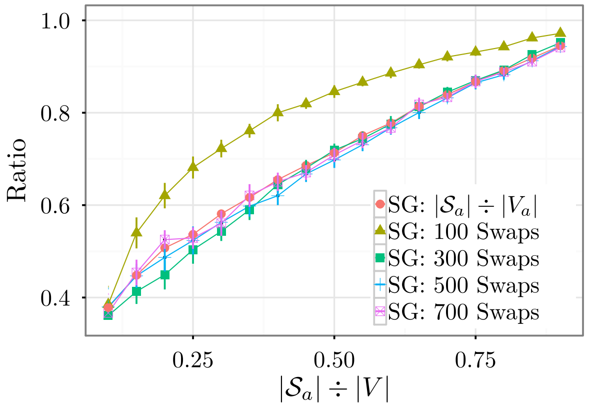

Fig. 17. Ratio of leaf subscribers over subscriber ratio, each after *swaps* = {100, 300, 500, 700} compared to the optimal ratio of subscribers for anonymity.

positions cool down early. Thus, many subscribers switch back and stay in leaf positions.

Fig. 17 illustrates the high sensitivity of SG against the subscriber ratio (cf. Fig. 9). In particular, an increase in the number of SGs by lowering λ (compare 300 SGs with 700 SGs) cannot sufficiently compensate. Therefore, the SG has to be combined with PF for higher subscriber ratios.

5.5. Probabilistic Forwarding & Shell Game (PFSG)

While SG provides protection against the global attacker for low ratios of subscribers, high ratios of overlay subscribers require PF in addition. For instance, if $|S|/|V|$ exceeds 0.25, 0.514 of all overlay nodes are subscribers (cf. Fig. 17). Therefore, we use PF to increase the ratio of overlay forwarders. The results in Fig. 12 give an indicator how to adjust μ to lower the ratio of overlay subscribers below 0.5.

5.6. Evaluation summary

In this section, we summarize simulation results regarding the effectiveness of the global attacker against AnonPubSub with PF and SG. Without these countermeasures, this attacker is highly effective in disclosing subscribers. The simulations indicate that PF cannot completely counter this attacker. We identified a linear relationship between PF parameter μ and loss in attack accuracy. Thus, no optimal parameter assignment exists, and the trade-off between loss of attack accuracy and signaling overhead has to be decided per use case.

The SG reduces the attack accuracy to the anonymity optimum. The SG works best for low ratios of subscribers as well as low node degree. The SG converges quickly towards an optimal mixture of leaf subscribers and forwarders. Therefore, the SG only requires moderate signaling overhead. In our setup, the decay function leads to a new balance between anonymity protection and signaling overhead. In scenarios where the SG does not perform well, the SG and PF can be combined to PFSG at the cost of an increased signaling overhead.

6. Conclusion

We present requirements for secure and anonymous pub-sub and complement the anonymity requirement with a strong and realistic attacker definition. We discuss building blocks to construct distributed and privacy-preserving pub-sub systems and analyze the related work. In particular, we identify that privacy-preserving pub-sub systems provide confidentiality but lack anonymity.

We present AnonPubSub for distributed and anonymous pub-sub. AnonPubSub establishes attribute overlays in a privacy-preserving manner. For that, participants obtain key material from a TTP depending on their role upon joining the system. Publishers then advertise their attributes by flooding information through the membership management. Subscribers establish the attribute overlay spanning subscribers, publishers, and forwarders based upon the advertisement, and thus form the attribute overlay. These overlays distribute notifications to subscribers with low latency and moderate overhead. Moreover, this construction protects publisher and subscriber anonymity against an internal attacker: publishers and subscribers remain concealed by forwarding messages via neighbors, and by omitting global identifiers.

We introduce two protective measures against the global attacker, namely Probabilistic Forwarding (PF) and the Shell Game (SG). PF pulls additional nodes into the overlay to hide subscribers, i.e., it increases the anonymity set. The SG shuffles the overlay topology and therefore prevents exposure of overlay properties to the global attacker.

We discuss how AnonPubSub meets the requirements for secure and anonymous pub-sub. Moreover, we complement the discussion with a quantitative analysis by simulating AnonPubSub, PF, and the SG. We confirm high effectiveness of the SG to protect anonymity against the global attacker, and identify scenarios where SG has to be complemented with PF.

In future work, we will elaborate on novel anonymity attack schemes for colluding internal adversaries in such overlay meshes. In addition, we will further improve integrity and availability. Hence we need to study attacks on the infrastructure and need to develop mechanism for a resilient attribute mesh construction without compromising anonymity.

Acknowledgment

This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) (No. B0101-15-1292, Development of Smart Space to promote the Immersive Screen Media Service), and by CASED (www.cased.de).

References

[1] M. Nabeel, S. Appel, E. Bertino, A. Buchmann, Privacy preserving context aware publish subscribe systems (extended version), in: Proceedings of NSS 2013, Springer, 2013, pp. 465–478, doi:10.1007/978-3-642-38631-2_34.
[2] M. Nabeel, N. Shang, B. Elisa, Efficient privacy preserving content based publish subscribe systems, in: Proceedings of SACMAT, ACM, 2012, pp. 133–144, doi:10.1145/2295136.2295164.
[3] A. Shikfa, M. Önen, R. Molva, Privacy and confidentiality in context-based and epidemic forwarding, Comput. Commun. 33 (13) (2010) 1493–1504, doi:10.1016/j.comcom.2010.04.035.
[4] M.A. Tariq, B. Koldehofe, A. Altaweel, K. Rothermel, Providing basic security mechanisms in broker-less publish/subscribe systems, in: Proceedings of DEBS, ACM, 2010, pp. 38–49, doi:10.1145/1827418.1827425.

[5] W. Chen, J. Jiangt, N. Skocik, On the privacy protection in publish/subscribe systems, in: Proceedings of WCNIS, IEEE, 2010, pp. 597–601, doi:10.1109/WCINS.2010.5541849.
[6] J. Daubert, M. Fischer, S. Schiffner, M. Mühlhäuser, Distributed and anonymous publish-subscribe, in: Proceedings of NSS, in: Volume 7873 of Lecture Notes in Computer Science, Springer, 2013, pp. 685–691, doi:10.1007/978-3-642-38631-2_57.
[7] C. Wang, A. Carzaniga, D. Evans, A.L. Wolf, Security issues and requirements for internet-scale publish-subscribe systems, in: Proceedings of HICSS, IEEE, 2002, pp. 3940–3947, doi:10.1109/HICSS.2002.994531.
[8] C. Raiciu, D.S. Rosenblum, Enabling confidentiality in content-based publish/subscribe infrastructures, in: Proceedings of SecureComm, IEEE, 2006, pp. 1–11, doi:10.1109/SECCOMW.2006.359552.
[9] S. Schiffner, S. Clauß, Using linkability information to attack mix-based anonymity services, in: Proceedings of PETS 2009, in: Volume 5672 of Lecture Notes in Computer Science, Springer, 2009, pp. 94–107, doi:10.1007/978-3-642-03168-7_6.
[10] A. Pfitzmann, M. Köhntopp, Anonymity, unobservability, and pseudonymity – a proposal for terminology, in: Proceedings of International Workshop on Design Issues in Anonymity and Unobservability, Springer, 2000, pp. 1–9, doi:10.1007/3-540-44702-4_1.
[11] G.D. Crescenzo, B.A. Coan, J.L. Schultz, S. Tsang, R.N. Wright, Privacy-preserving publish/subscribe: efficient protocols in a distributed model, in: Proceedings of DPM, SETOP 2013, Egham, UK, September 12–13, 2013, Revised Selected Papers, in: Volume 8247 of Lecture Notes in Computer Science, Springer, 2013, pp. 114–132, doi:10.1007/978-3-642-54568-9_8.
[12] G. Danezis, Statistical disclosure attacks, in: Proceedings of SEC, in: Volume 250 of IFIP Conference Proceedings, Kluwer, 2003, pp. 421–426.
[13] G. Danezis, C. Díaz, C. Troncoso, Two-sided statistical disclosure attack, in: Proceedings of PETS, in: Volume 4776 of Lecture Notes in Computer Science, Springer, 2007, pp. 30–44, doi:10.1007/978-3-540-75551-7_3.
[14] C. Troncoso, B. Gierlichs, B. Preneel, I. Verbauwhede, Perfect matching disclosure attacks, in: Proceedings of PETS, Springer, 2008, pp. 2–23, doi:10.1007/978-3-540-70630-4_2.
[15] P.T. Eugster, P. Felber, R. Guerraoui, A. Kermarrec, The many faces of publish/subscribe, ACM Comput. Surv. 35 (2) (2003) 114–131, doi:10.1145/857076.857078.
[16] A.J. Ganesh, A. Kermarrec, L. Massoulié, Peer-to-peer membership management for gossip-based protocols, IEEE Trans. Comput. 52 (2) (2003) 139–149, doi:10.1109/TC.2003.1176982.
[17] A.I.T. Rowstron, A. Kermarrec, M. Castro, P. Druschel, SCRIBE: the design of a large-scale event notification infrastructure, in: Proceedings of NGC 2001, COST264 Workshop, in: Volume 2233 of Lecture Notes in Computer Science, Springer, 2001, pp. 30–43, doi:10.1007/3-540-45546-9_3.
[18] I. Clarke, O. Sandberg, B. Wiley, T.W. Hong, Freenet: a distributed anonymous information storage and retrieval system, in: Proceedings of Designing Privacy Enhancing Technologies, International Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, USA, July 25–26, 2000, in: Volume 2009 of Lecture Notes in Computer Science, Springer, 2000, pp. 46–66, doi:10.1007/3-540-44702-4_4.
[19] Q. Lv, P. Cao, E. Cohen, K. Li, S. Shenker, Search and replication in unstructured peer-to-peer networks, in: Proceedings of SIGMETRICS 2002, June 15–19, 2002, Marina Del Rey, California, USA, ACM, 2002, pp. 258–259, doi:10.1145/511334.511369.
[20] J. Khoury, G. Lauer, P.P. Pal, B. Thapa, J.P. Loyall, Efficient private publish-subscribe systems, in: Proceedings of ISORC 2014, Reno, NV, USA, June 10–12, 2014, IEEE, 2014, pp. 64–71, doi:10.1109/ISORC.2014.10.
[21] S. Rafaeli, D. Hutchison, A survey of key management for secure group communication, ACM Comput. Surv. 35 (3) (2003) 309–329, doi:10.1145/937503.937506.
[22] A. Lysyanskaya, R.L. Rivest, A. Sahai, S. Wolf, Pseudonym systems, in: Selected Areas in Cryptography, Springer, 1999, pp. 184–199, doi:10.1007/3-540-46513-8-14.
[23] M. Srivatsa, L. Liu, Securing publish-subscribe overlay services with EventGuard, in: Proceedings of CCS, ACM, 2005, p. 289, doi:10.1145/1102120.1102158.
[24] D. Mhapasekar, Accomplishing anonymity in peer to peer network, in: Proceedings of ICCCS, ACM, 2011, pp. 555–558, doi:10.1145/1947940.1948055.
[25] A. Shikfa, M. Onen, R. Molva, Broker-based private matching, in: Privacy Enhancing Technologies (PETS), Springer, Waterloo, Canada, 2011, pp. 264–284, doi:10.1007/978-3-642-22263-4_15.
[26] R. Barazzutti, P. Felber, Thrifty privacy: efficient support for privacy-preserving publish/subscribe, in: Proceedings of DEBS, ACM, 2012, pp. 225–236, doi:10.1145/2335484.2335509.
[27] M. Khambatti, K.D. Ryu, P. Dasgupta, Structuring peer-to-peer networks using interest-based communities, in: Proceedings of DBISP2P, in: Volume 2944 of Lecture Notes in Computer Science, Springer, 2003, pp. 48–63, doi:10.1007/978-3-540-24629-9_5.
[28] A. Shikfa, M. Onen, R. Molva, Privacy in context-based and epidemic forwarding, in: Proceedings of WoWMoM, IEEE, 2009, pp. 1–7, doi:10.1109/WOWMOM.2009.5282445.
[29] M.A. Tariq, G.G. Koch, B. Koldehofe, I. Khan, K. Rothermel, Dynamic publish/subscribe to meet subscriber-defined delay and bandwidth constraints, in: Proceedings of Euro-Par 2010, in: Volume 6271 of Lecture Notes in Computer Science, Springer, 2010, pp. 458–470, doi:10.1007/978-383-642-15277-1_44.
[30] R. Chandra, V. Ramasubramanian, K.P. Birman, Anonymous gossip: Improving multicast reliability in mobile ad-hoc networks, in: Proceedings of ICDCS 2001, IEEE, 2001, pp. 275–283, doi:10.1109/ICDSC.2001.918957.
[31] P.F. Syverson, D.M. Goldschlag, M.G. Reed, Anonymous connections and onion routing, in: Proceedings of IEEE S&P, IEEE Computer Society, 1997, pp. 44–54, doi:10.1109/SECPRI.1997.601314.
[32] A. Shikfa, M. Önen, R. Molva, Privacy-preserving content-based publish/subscribe networks, in: Proceedings of IFIP SEC, Springer, 2009, pp. 270–282, doi:10.1007/978-3-642-01244-0_24.
[33] D.M. Goldschlag, M.G. Reed, P.F. Syverson, Onion routing, Commun. ACM 42 (1999) 39–41, doi:10.1145/293411.293443.
[34] J. Gray, Notes on data base operating systems, in: Operating Systems, in: Volume 60 of Lecture Notes in Computer Science, Springer, 1978, pp. 393–481.
[35] P.A. Bernstein, V. Hadzilacos, N. Goodman, Concurrency Control and Recovery in Database Systems, 370, Addison-wesley, New York, 1987.
[36] O. Berthold, H. Langos, Dummy traffic against long term intersection attacks, in: Proceedings of PET, in: Volume 2482 of Lecture Notes in Computer Science, Springer, 2002, pp. 110–128, doi:10.1007/3-540-36467-6_9.
[37] D.J. Watts, S.H. Strogatz, Collective dynamics of 'small-world'networks, Nature 393 (6684) (1998) 440–442, doi:10.1038/30918.