Distributed and Anonymous Publish-Subscribe

Jörg Daubert, Mathias Fischer, Stefan Schiffner, and Max Mühlhäuser

Technische Universität Darmstadt, CASED,
Telecooperation Group, Hochschulstraße 10, 64283 Darmstadt
{joerg.daubert,mathias.fischer,stefan.schiffner}@cased.de,
max@informatik.tu-darmstadt.de

Abstract. Publish-subscribe is a scheme for distributing information based on interests. While security mechanisms have been added to publish-subscribe, privacy, in particular anonymous communication is hardly considered. We summarize security and privacy requirements for such systems, including an adversary model for privacy. We introduce a construction for publish-subscribe overlays that fulfills the requirements. Contrary to previous approaches, it does neither presume an online trusted third party, nor expensive cryptographic operations performed by brokers. Further, we informally discuss how our requirements are met.

Keywords: privacy, pub-sub, overlay.

1 Introduction

Publish-subscribe decouples producers (*publishers*) and consumers (*subscribers*) of information by introducing super nodes, the *brokers*. Subscribers announce their interests to the broker (subscription), while publishers send information (notification) to the broker. Brokers match and distribute notifications. Privacy is desirable, e.g., in private car sharing, dating services, or citizen journalism. In the latter, participants publish and consume news, and might be subject to repression, e.g., whistleblowers and politically prosecuted people. As a result, a publish-subscribe (pub-sub) system intended for a deployment in such a scenario has to fulfill several requirements to protect its users. We require a pub-sub system to comply with anonymity, confidentiality, scalability, integrity, authenticity, and availability: participants are anonymous w.r.t. an adversary if they are unidentifiable for the adversary within a set of participants, the anonymity set [9]. Information must be transmitted secretly between sender and receiver. The system must remain scalable in terms of number of supported nodes. Alteration of a message must be detectable by the receiver (integrity). Only authorized participants can send authentic notifications and can read notifications. The system must maintain availability in the presence of node failures and attacks.

We define privacy in pub-sub as the combination of participant anonymity and confidentiality. Both requirements are closely related, as the lack of one can lead to a violation of the other one [11]. Privacy adversaries can be structured w.r.t. their capabilities: a *passive* adversary only observes messages, while an

J. Lopez, X. Huang, and R. Sandhu (Eds.): NSS 2013, LNCS 7873, pp. 685–691, 2013.

active one can alter. An adversary with knowledge about pseudonyms or keys is an *insider*. Furthermore, it can have either *global* or *local* topology knowledge. Finally, the internal adversary can collude with other internal adversaries. We focus on an active insider with full topology information on the communication network, but it can only observe its own communication channels. It is strong as it can act adaptively, exploit the topology information, and force the system to react by sending valid messages.

A pub-sub system complies with subscriber anonymity, or publisher anonymity respectively, w.r.t. an adversary and an attribute, if the subscriber cannot be identified within the anonymity set. A pub-sub system complies with notification confidentiality, or subscription confidentiality respectively, w.r.t. an adversary, if the adversary does not learn the attribute, the notification, or the subscription content.

Several approaches for realizing privacy-preserving pub-sub schemes exist [1–3, 7, 8, 10, 12–15]. Most approaches focus on confidentiality [1–3, 7, 8, 10, 12, 14], but do not consider anonymity [16]. To ensure confidentiality, many contributions encrypt information [2, 3, 7, 8, 10, 12, 15], leverage a private matching scheme [1–3, 7, 8, 10, 12], and describe key management [12, 14, 15]. However, approaches [2, 8] assume an out-of-band key exchange. Moreover, approaches [2, 3, 8, 13] even require the knowledge of the cryptographic keys of participants. This violates anonymity as well as space decoupling, a functional pub-sub requirements that decouples publishers from subscribers. Approaches [10, 14] provide space decoupling, but depend on a central, online Trusted Third Party (TTP), which is a Single Point of Failure (SPoF) that violates availability and scalability. The key exchange protocols in [2, 3, 13] do not scale with the number of participants. Approaches [3, 12] only consider honest but curious brokers, but not malicious publishers and subscribers. Further, [15] is fully distributed, and therefore scalable as well as free of SPoFs, but does not sufficiently describe the membership management. Finally, multicast protocols, e.g., PIM-SM [4], can be used to establish distribution trees and scale well, but do not provide anonymity and confidentiality.

Summarizing the related work, none of the articles provides anonymity to both—publisher and subscriber—and scalability. Therefore, none of those systems achieves the listed requirements—most dominantly not anonymity, confidentiality and scalability at once.

In this paper, we bridge the gap between privacy by means of anonymity and confidentiality, as well as scalability for pub-sub systems. The main contribution of our article is a scalable, anonymous and confidential method for pub-sub overlay construction, which is described in the following Section. Moreover, we discuss our solution along these requirements and the attacker model in Section 3. Finally, Section 4 concludes the article.

2 Privacy-Preserving and Scalable pub-sub Overlay

We create pub-sub overlay topologies that enable efficient privacy-preserving notification distribution from publishers to subscribers. We assume a connected

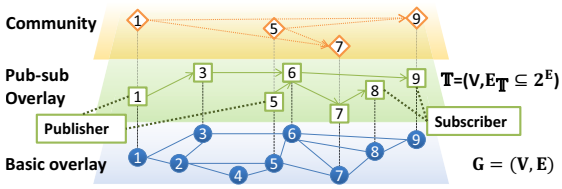

Fig. 1. Basic overlay (bottom), pub-sub overlay (middle), community (top)

basic overlay network G (lowest layer of Fig. 1), e.g., established via SCAMP [6]. This basic overlay provides each node with a neighborhood set, as well as confidential links between neighbors. On top, we construct a mesh network $\mathcal{M}_a$ per attribute a. Each mesh spans a subset of overlay members (middle layer of Fig. 1). The pub-sub overlay is a union of all meshes. The top layer in Fig. 1 depicts the desired notification flow. To establish mesh $\mathcal{M}_a$, we use advertisements and subscriptions prior to sending notifications. Though, we present the overlay construction with a single attribute, it is generalizable to multiple attributes.

We use the following notations: the set of publishers $\mathcal{P}$, subscribers $\mathcal{S}$, forwarders $\mathcal{F}$, and $\mathcal{S}_a \subseteq \mathcal{S}$ for the set of subscribers interested in an attribute $a \in \mathcal{A}$. Respectively, $\mathcal{P}_a \subseteq \mathcal{P}$ denotes the set of publishers publishing attribute a. The graph $G := (V, E)$ as basic overlay with participants $V := \mathcal{P} \cup \mathcal{S} \cup \mathcal{F}$ and edges E. Further, meshes $\mathcal{M}_a := (V_{\mathcal{M}} \subseteq V, E_{\mathcal{M}} \subseteq E)$ as subgraphs in overlay $\mathbb{T}$ for an attribute a and a subset of edges $E_{\mathcal{M}} \subseteq E$.

2.1 Overlay Construction

The overlay construction is split in an advertisement phase and a subscription phase. We use an offline TTP for the initial distribution of key material, which is used for confidentiality and authenticity. Confidentiality is achieved by using symmetric keys. Authenticity and message integrity are achieved using signatures and certificates issued by the TTP. Each participant initially connects to the TTP and presents its desired attributes. If the TTP grants access to the participant, the TTP releases the corresponding key material.

Cryptographic Primitives. The *TTP* owns a secret key sk_{TTP} and a public key pk_{TTP}. Furthermore, we use the following functions: $keyGen(\mathrm{a}) \mapsto (sk_a, pk_a, K_a)$ generates a key triple: the signature key sk_a for attribute a, the signature verification key pk_a, and the symmetric key K_a. $enc(m, K) \mapsto \{m\}_K$ encrypts message m with key K to cipher text $\{m\}_K$. $sign(m, sk) \mapsto m^{sk}_{sig}$ signs a message m using a cryptographic hash function and the private key sk and creates the signature m^{sk}_{sig}. $cert(pk, t, sk_{TTP}) \mapsto cert_{pk}$ returns $cert_{pk} = (pk, t, sign(pk||t, sk_{TTP}))$, a certificate for public key pk and an associated token t. Here, $pk||t$ denotes the concatenation of pk and t. Function $verify(m, m^{sk}_{sig}, pk) \mapsto true$ or $false$ uses pk to verify that signature m^{sk}_{sig} is a valid signature of m that was created with the secret key sk.

The *TTP* prepares keys and certificates: for each attribute $a \in \mathcal{A}$, it executes $keyGen_A$ and stores the output triple. For certificates, the *TTP* obtains $t_a = enc(a, K_a)$, creates $cert_a$ by executing $cert(pk_a, t_a, sk_{TTP})$, and stores it together with the triple.

Advertisement Phase. We spread the information about attributes and where to obtain them in the basic overlay G. Further, we have to ensure that only legitimate subscribers for a are able to link a to the message due to notification confidentiality. For that, every new publisher need to contact the *TTP* to retrieve the keys for a: $(sk_a, pk_a, K_a, cert_a)$. Every new subscriber just obtains (K_a).

Each advertisement for attribute a contains a token t_a, so that subscribers can match their interest against it. Thus, only legitimate subscribers in possession of the respective K_a can link t_a to a. To spread the information about a, a publisher floods G by sending advertisements to its neighbors in G. As t_a is identical for all publishers in possession of the same a, every forwarder can suppress duplicates.

To prevent overlay partitioning, forwarders must be able to distinguish duplicates from the same publisher and two different publishers $p_1, p_2 \in \mathcal{P}_a$. However, publisher IDs contradict anonymity. Random time-to-live counters would allow adversaries to lie. This would enable them to capture more messages and to partition the overlay.

To overcome this problem, we use hash chains. They serve as transaction pseudonyms per advertisement. Hash chains allow the selection of the shortest path, while preventing adversaries from lying. Given a cryptographic hash function $H(h) = h'$ that takes h as input and outputs h', a hash chain is defined by all values h_i derived by repeatedly applying H to its output. The function *isChain* tests if two hash values h_1 and h_2 belong to the same hash chain and have a maximum distance d_{max}:

$$isChain : (h_1, h_2) \mapsto \{true, false\} :$$
$$\forall_{i \in \{0..d_{max}\}} : if\, H^{(i)}(h) \stackrel{?}{=} h' \vee H^{(i)}(h') \stackrel{?}{=} h \mapsto true, \text{ otherwise } false$$

The parameter d_{max} is equivalent to a time-to-live and has to be set according to the expected maximum diameter of G. The publisher generates a random h from the output domain of H and attaches it together with token t_a to an advertisement message (t_a, h). Forwarders keep routing tables, containing triples (t_a, h, v), where v is the neighbor from which the advertisement has been received from. If a forwarder f has not received token t_a before, f stores it in the routing table as (t_a, h, v) and forwards (t_a, h') with $h' := H(h)$ to all other neighbors. Otherwise, f already has a triple (t_a, h_2, v_2). Then it checks if the output of $isChain(h, h_2)$ returns *true*: hence, both advertisements belong to the same hash chain. Further, in case h is a predecessor of h_2 in the hash chain, a shorter path has been discovered and f replaces (t_a, h_2, v_2) by (t_a, h, v). If $isChain(h, h_2)$ returns *false*, h and h_2 belong to different hash chains. Thus, another publisher for the same t_a has been found, and f stores (t_a, h, v) in the routing table.

Tokens must be signed, so that an adversary cannot create arbitrary advertisements and flood G. For that, the publisher executes $sign(t_a, sk_a)$ and obtains $t_a{}^{sk_a}_{sig}$.

The advertisement is then extended to message $(t_a, t_a{}^{sk_a}_{sig}, cert_a, h)$. Every participant verifies advertisements by checking $cert_a$ and $t_a{}^{sk_a}_{sig}$. Hence, participants can detect non-authentic advertisements as well as duplicates.

Subscription Phase. Subscribers identify advertisements of interest, and establish distribution paths for notifications via subscription messages. The result is an overlay mesh $\mathcal{M}_a$.

For that, subscribers compare the token t_x from an advertisement triple (t_x, h', v) to their attributes of interests. A subscriber encrypts each own a with the corresponding key K_a and compares the result t_a with t_x. Once the t_x from the triple is confirmed to match t_a, the subscriber s joins the mesh $\mathcal{M}_a$ via a subscription.

Subscriptions are sent back the reverse path of the advertisements. A subscriber adds tuple (t_a, s) to the subscription table, where s is the subscribing node itself. Moreover, it sends a subscription message (t_a) towards the origin of the advertisement. Whenever a forwarder f receives a subscription (t_a) from neighbor v, f updates its subscription table with the tuple (t_a, v). If there is no subscription entry (t_a, v_x) for any neighbor v_x, f forwards the subscription. For that, f retrieves all records (t_a, h_m, v_m) matching t_a from the routing table and sends subscription (t_a) to each neighbor v_m.

In case there are multiple publishers for the same attribute, additional measures are required to ensure mesh $\mathcal{M}_a$ is connected. Let us assume two publishers p_1 and p_2, and p_2 joins after p_1. Then p_2 receives an advertisement from p_1 via a neighbor v. Now p_2 subscribes towards p_1 via v, and therefore establishes a directed connection between p_1 and p_2 in $\mathcal{M}_a$. In case a node leaves the system, the remaining nodes repair the mesh $\mathcal{M}_a$ with unsubscribe and unadvertise messages.

2.2 Content Distribution

After securely establishing the distribution overlays M_a per attribute a, notifications need to be transported. A notification for attribute a originated by a publisher p contains token t_a as routing identifier and some content m. The notification is flooded in the pre-established mesh $\mathcal{M}_a$. For that, p, and every subsequent forwarder, looks up all records matching t_a in its subscription and routing tables, and sends the notification to each v_x, except the one the notification was received from.

While t_a does not leak any plaintext information, m is accessible by every traversed node. Hence, m must be protected to ensure notification confidentiality. Both roles, $\mathcal{P}_a$ and $\mathcal{S}_a$, share a symmetric key K_a. Publisher p encrypts m using K_a and obtains $\{m\}_{K_a}$. Hence, a notification $(t_a, \{m\}_{K_a})$ does not leak any more information than an advertisement. Finally, the same signature scheme as for advertisements is applied to obtain an authentic notification $(t_a, \{m\}_{K_a}, sig)$, with $sig = sign(t_a || \{m\}_{K_a}, sk_a)$.

3 Discussion

A privacy-preserving pub-sub system has to comply with the requirements defined in Section 1 to be applicable to the citizen journalism scenario. We analyze our pub-sub overlay with respect to these requirements. We assume the TTP to be honest w.r.t. not disclosing information.

Anonymity. Our system provides publisher and subscriber anonymity. The adversary has full topology information on G. For subscriber anonymity, we assume a single adversarial publisher that publishes one attribute. Hence, the distribution mesh is a tree and the adversary is the root. As subscriptions are merged by branch nodes, at most one subscription per branch reaches the adversary. Thus, the adversary cannot distinguish subscribers from other nodes within each branch. Hence, the anonymity set size of a subscriber in a branch is the size of this branch. For publisher anonymity, we assume a single adversarial subscriber that subscribes only to one attribute. Further, the advertisement is received via the shortest path first. With G, the adversary can construct a tree from all shortest paths starting on its own node to all nodes. Hence, the anonymity set size of a publisher in a branch is the size of this branch.

Confidentiality. Nodes that are neither publisher nor subscriber cannot decipher messages. We use a symmetric crypto-system, which provides security to known plaintext attacks for end-to-end encryption. Still, due to decoupling in pub-sub [5], if a notification, or advertisement respectively, matches a subscription or not can be always learned [10] by observation.

Integrity and Authenticity. Our system provides message integrity and authenticity for advertisement and notifications by using digital signatures. However, subscriptions are not authentic. Moreover, an adversary in possession of a corresponding key may alter foreign advertisements and notifications.

Scalability A scalable system grows at most proportionally with the number of participants in terms of resources per participant. The required node memory for the basic overlay depends on the size of its neighborhood and remains constant [6]. Hence, the required memory for the overlay grows proportionally with the number of attributes.

Resilience. A robust system can compensate node failure or provide at least graceful degradation. The presented pub-sub construction method represents a Peer-to-Peer (P2P) network without online SPoFs and bottlenecks, as each peer can take over the role of the other ones.

4 Conclusion

We presented the citizen journalism scenario, stated security requirements for pub-sub systems in such a scenario, and categorized privacy adversaries. Further,

we analyzed related approaches, and introduced our complementary construction to privacy-preserving pub-sub overlays. Finally, we discussed our construction w.r.t. the requirements. The construction protects participant anonymity, keeps information confidential, scales, and does not depend on central structures except the offline TTPs. Therefore, our work is applicable to citizen journalism. Future work will reduce the dependency on the offline TTPs and study the anonymity of different overlay types.

References

1. Barazzutti, R., Felber, P., et al.: Thrifty Privacy: Efficient Support for Privacy-Preserving Publish / Subscribe. In: DEBS, pp. 225–236. ACM (2012)
2. Chen, W., Jiangt, J., Skocik, N.: On the privacy protection in publish/subscribe systems. In: WCNIS, pp. 597–601. IEEE (2010)
3. Choi, S., Ghinita, G., Bertino, E.: A Privacy-Enhancing Content-Based Publish/Subscribe System Using Scalar Product Preserving Transformations. In: Bringas, P.G., Hameurlain, A., Quirchmayr, G. (eds.) DEXA 2010, Part I. LNCS, vol. 6261, pp. 368–384. Springer, Heidelberg (2010)
4. Fenner, B., et al.: Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised). RFC 4601 (Proposed Standard) (2006)
5. Eugster, P.T., Felber, P., Guerraoui, R., Kermarrec, A.: The many faces of publish/subscribe. ACM Computing Surveys (CSUR) 35(2), 114–131 (2003)
6. Ganesh, A.J., Kermarrec, A., Massoulié, L.: Peer-to-Peer Membership Management for Gossip-Based Protocols. IEEE (TC) 52(2), 139–149 (2003)
7. Ion, M., Russello, G., Crispo, B.: Supporting Publication and Subscription Confidentiality in Pub/Sub Networks. In: Jajodia, S., Zhou, J. (eds.) SecureComm 2010. LNICS, vol. 50, pp. 272–289. Springer, Heidelberg (2010)
8. Nabeel, M., Shang, N., Elisa, B.: Efficient privacy preserving content based publish subscribe systems. In: SACMAT, pp. 133–144. ACM (2012)
9. Pfitzmann, A., Köhntopp, M.: Anonymity, Unobservability, and Pseudonymity - A Proposal for Terminology. In: Federrath, H. (ed.) Anonymity 2000. LNCS, vol. 2009, pp. 1–9. Springer, Heidelberg (2001)
10. Raiciu, C., Rosenblum, D.S.: Enabling Confidentiality in Content-Based Publish/Subscribe Infrastructures. In: SecureComm, pp. 1–11. IEEE (August 2006)
11. Schiffner, S., Clauß, S.: Using linkability information to attack mix-based anonymity services. In: Goldberg, I., Atallah, M.J. (eds.) PETS 2009. LNCS, vol. 5672, pp. 94–107. Springer, Heidelberg (2009)
12. Shikfa, A., Önen, M., Molva, R.: Privacy in context-based and epidemic forwarding. In: WoWMoM, pp. 1–7. IEEE (June 2009)
13. Shikfa, A., Önen, M., Molva, R.: Privacy-Preserving Content-Based Publish/Subscribe Networks. In: Gritzalis, D., Lopez, J. (eds.) SEC 2009. IFIP AICT, vol. 297, pp. 270–282. Springer, Heidelberg (2009)
14. Srivatsa, M., Liu, L.: Securing publish-subscribe overlay services with EventGuard. In: CCS, p. 289. ACM (2005)
15. Tariq, M.A., Koldehofe, B., Altaweel, A., Rothermel, K.: Providing basic security mechanisms in broker-less publish / subscribe systems. In: DEBS, pp. 38–49. ACM (July 2010)
16. Wang, C., Carzaniga, A., Evans, D., Wolf, A.L.: Security Issues and Requirements for Internet-Scale Publish-Subscribe Systems. In: HICSS, pp. 3940–3947. IEEE (2002)