# Towards Scalable and Dependable Privacy-Preserving Publish/Subscribe Services

Emanuel Onica, Pascal Felber, Hugues Mercier, Etienne Rivière

# Towards Scalable and Dependable Privacy-Preserving Publish/Subscribe Services

Emanuel Onica
Alexandru Ioan Cuza University of Iaşi, Romania
Email: eonica@info.uaic.ro

Pascal Felber, Hugues Mercier and Etienne Rivière
University of Neuchâtel, Switzerland
Email: first.last@unine.ch

***Abstract*—Security guarantees are required features for cloud services. Among these, the preservation of user data privacy is critical. However, the necessity to perform computation over user data by the service providers makes this guarantee hard to provide. This necessity often becomes a requirement for services that depend on the nature of the data, such as publish/subscribe (pub/sub) routing solutions. Most of proposed solutions for privacy-preserving pub/sub fail to satisfactorily address scalability and dependability along privacy issues. In this paper, we define a set of requirements that should be met by a scalable and dependable privacy-preserving pub/sub solution.**

## I. Introduction

Preserving data privacy is a critical security requirement for cloud based services that handle user data. Publish/subscribe (pub/sub) [1] is an information dissemination model that matches well the needs of complex and multi-party cloud computation. Pub/sub decouples data publishers form consumers in a simple and intuitive fashion. A publication data stream contains information of interest to users who register subscriptions to the cloud service. The service uses dedicated machines, referred as brokers, to store the subscriptions, perform the matching of incoming publications and notify the users matching accordingly. Users are increasingly concerned by the unauthorized access to their private data. The content of subscriptions and publications unfortunately reveals such private data. Various solutions have been proposed that permit encrypting subscriptions and publications so that the untrusted cloud service can still perform the matching. Most of these solutions require a complementary secure key exchange between the subscribers and the publisher of the data stream. This key exchange is typically not defined or disregards the scalability and dependability aspects.

## II. Service architecture requirements

We define a set of key requirements for a scalable and dependable privacy-preserving pub/sub service. We focus specifically on missing features in existing solutions.

**Core privacy requirement:** *possibility to perform comprehensive encrypted matching* for the cloud service provider.

We consider as core requirement the capability by the cloud service to match an encrypted publication over encrypted subscriptions, which we name *encrypted matching*. Many privacy-preserving solutions do not offer this feature (e.g., [2], [3]) and typically rely on end-to-end encryption of sensitive data and access control separation. In this setting the pub/sub service is not able to perform matching over the encrypted data, but only on the non-encrypted part, which severely limits areas of application (e.g., one cannot use subscription fields for routing, and sensitive publications are simply not routable). An extra desirable characteristic is the possibility to perform encrypted matching over ranged subscription constraints (e.g., determining if a stock value is higher than a certain threshold). This is a limitation of some of the current solutions (e.g., [4], [5]) which only allow matching over equality constraints, preventing their application to full-fledged content-based pub/sub services.

**Requirement 1:** *grouping hosts in security domains* for individual subscribers, publishers and pub/sub service brokers.

In order for the pub/sub service to perform encrypted matching, publishers and subscribers must use a common or correlated encryption key. The decoupled nature of pub/sub communication is a problem in this context: since there is no knowledge on the publisher side about the receiving subscribers, the pub/sub system ignores to which publisher/subscriber pairs the key must be distributed. Although it seems difficult to overcome this obstacle without losing the benefits offered by the pub/sub paradigm, for many applications there are relations between the security domains of publishers and subscribers. This permits creating groups of hosts with the same level of trust that can exchange common keys. Consider as an example a user in a financial company that registers a subscription for stock exchange information. This subscriber does not know the exact host that emits publications, but the financial company does know from which stock exchange it wants to obtain the publication data. Likewise, the stock exchange might charge fees for providing the data stream to all of the subscribers from a given financial company.

**Requirement 2:** *support for subscription re-encryption* in the pub/sub service by the encrypted matching scheme.

The key used to encrypt the pub/sub traffic must be periodically updated. A trigger for this is the eviction of a corrupted subscriber sharing a group key with remaining subscribers. Another reason is to counteract brute force attacks meant to guess the current key in use. A problem that appears when the encryption key is updated is the invalidation of subscriptions stored by the untrusted pub/sub service. These subscriptions remain encrypted with the old invalid key, and since publishers will use the new key the pub/sub service will no longer be able to perform the encrypted matching. A naive way to solve this problem is to notify users to re-encrypt and re-register their former subscriptions using the new key, however this approach has major performance and dependability drawbacks. First, while the subscriptions are re-encrypted and re-registered the pub/sub service will fail to deliver matching publications. Second, if the service provides reliable storage guarantees to users for their subscriptions, these guarantees are moot because

users must locally store a copy of their subscriptions in order to re-encrypt them. Third, a scalability issue can also appear due to high demand in network usage (all subscriptions encrypted with the former key by all users must be re-registered at once).

We have noticed and addressed these issues for the first time in [6], where we extended an encrypted matching scheme [7] to allow the untrusted pub/sub service to directly re-encrypt the stored subscriptions when a new key is exchanged. This is done by giving a secure token to the untrusted pub/sub service. In the following we generically formalize this feature. We believe this can be used for defining subscription re-encryption support for other schemes (e.g., [8]) allowing encrypted matching. Let us consider $E_K(S)$, the ciphertext obtained by encrypting subscription $S$ with key $K$ using an encrypted matching algorithm $E$. The support for subscription re-encryption implies that for two keys $K_1$ and $K_2$, a function $K_R = f(K_1, K_2)$ can be defined, such that for an encrypted subscription $E_{K_1}(S)$ there exists a transformation $\tau(E_{K_1}(S), K_R) = E_{K_2}(S)$. $K_R$ is then the token that is given to the untrusted pub/sub service to perform the in-place re-encryption. Making $K_R$ available in an untrusted domain must not disclose any information about the encryption keys $K_1$ and $K_2$ nor the subscription plaintext $S$.

**Requirement 3:** *a scalable secure group communication key exchange protocol* adapted to the pub/sub domain grouping.

A protocol is required to disseminate common encryption keys to different security domains, where hosts reside. This preserves a partial decoupling of the individual hosts, keeping only a loose coupling at the domain level. Hosts used by the pub/sub service are typically part of an untrusted security domain, where keys must not be distributed. However, the protocol should support disseminating key update tokens defined in Requirement 3 to the pub/sub service domain.

Some solutions [5] ignore pub/sub decoupling and require subscribers to exchange key material directly with publishers for every subscription. Such designs have serious scalability issues. The idea of adapting a partially decoupled group key exchange protocol for pub/sub systems was proposed in [2] with OFT [9]. However, the work did not consider encrypted matching and dissemination of key update tokens to the pub/sub service. In [10] the authors suggest ELK [11] as a potential solution but do not develop the idea.

In [6] we included a simple hierarchical key exchange architecture for the single purpose of testing the subscription re-encryption. This proved scalable as we did not observe any significant increase in the key distribution time when increasing the number of brokers from 5 to 15. However, since the purpose of our work was not to test key exchange communication, we did not evaluate our protocol for large groups of subscriber hosts and we considered only three security domains.

**Requirement 4:** *continuity of service* during a key update.

The stored subscription invalidation is a problem that occurs when a key is updated whether or not subscription re-encryption is supported by the pub/sub service (the effects are of course minimized when the encrypted matching scheme supports the re-encryption). For a short duration, the publication stream matching may be stopped by the incompatibility between encryption keys, and it is critical to prevent this to happen. In [6] we used a transitional state keeping old subscriptions active for a custom defined period of time. We believe that a more thorough investigation could be done to address this requirement.

**Requirement 5:** *key exchange reliability* in case of failures.

Since the communication in the pub/sub system is dependent on the key exchange, it is imperative to extend the dependability guarantees to its implementation. It is useless to tolerate failures at the pub/sub service level if the unhandled crash of one of the key exchange dedicated hosts disables the service. We are not aware of any dedicated solution to this problem. An approach is to rely on third party solutions such as ZooKeeper's [12] fault tolerance support when implementing the key exchange.

## III. Conclusion

To the best of our knowledge, no privacy-preserving pub/sub solution considers all the requirements defined in this article. We consider that these requirements should be used as guidelines for future practical solutions. Our attempt towards scalable and dependable privacy-preserving pub/sub [6] fulfills the Core privacy requirement as well as Requirements 1 and 2. We also made steps to address at a basic level the other three requirements, on which we intend to focus our future work.

## Acknowledgment

This work is partly funded from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 692178.

## References

[1] P. T. Eugster, P. Felber, R. Guerraoui, and A.-M. Kermarrec, "The many faces of publish/subscribe," *ACM Computing Survey*, vol. 35, no. 2, 2003.

[2] J. Bacon, D. M. Eyers, J. Singh, and P. R. Pietzuch, "Access control in publish/subscribe systems," in *DEBS 2008: 2nd International Conference on Distributed Event-Based Systems*.

[3] Y. Zhao and D. Sturman, "Dynamic access control in a content-based publish/subscribe system with delivery guarantees," in *ICDCS 2006: 26th IEEE Intl. Conf. on Distributed Computing Systems*.

[4] P. Pal, G. Lauer, J. Khoury, N. Hoff, and J. Loyall, "P3S: a privacy preserving publish-subscribe middleware," in *13th ACM International Middleware Conference 2012*.

[5] G. Crescenzo, J. Burns, B. Coan, J. Schultz, J. Stanton, S. Tsang, and R. N. Wright, "Efficient and private three-party publish/subscribe," in *NSS 2013: : 7th Intl. Conference on Network and System Security*.

[6] E. Onica, P. Felber, H. Mercier, and E. Rivière, "Efficient key updates through subscription re-encryption for privacy-preserving publish/subscribe," in *16th ACM Intl. Middleware Conference 2015*.

[7] S. Choi, G. Ghinita, and E. Bertino, "A privacy-enhancing content-based publish/subscribe system using scalar product preserving transformations," in *DEXA 2010: Database and Expert Systems Applications*.

[8] H. Xu, S. Guo, and K. Chen, "Building confidential and efficient query services in the cloud with RASP data perturbation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 2, 2014.

[9] A. T. Sherman and D. A. McGrew, "Key establishment in large dynamic groups using one-way function trees," *IEEE Transactions on Software Engineering*, vol. 29, no. 5, 2003.

[10] J. Li, C. Lu, and W. Shi, "An efficient scheme for preserving confidentiality in content-based publish/subscribe systems," Georgia Institute of Technology, Tech. Rep. GIT-CC-04-01, 2004.

[11] A. Perrig, D. Song, and J. D. Tygar, "ELK, a new protocol for efficient large-group key distribution," in *SP 2001: IEEE Symposium on Security and Privacy*.

[12] P. Hunt, M. Konar, F. P. Junqueira, and B. Reed, "ZooKeeper: Wait-free coordination for internet-scale systems," in *ATC 2010: USENIX Annual Technical Conference*.