Identity Management and Integrity Protection in Publish-Subscribe Systems

Anders Fongen, Federico Mancini

Identity Management and Integrity Protection in Publish-Subscribe systems

Anders Fongen and Federico Mancini

Norwegian Defence Research Establishment
anders.fongen@ffi.no

Abstract. The use of Identity Management (IdM) may leverage the trust in a distributed Publish-Subscribe (PubSub) system. An IdM provides mutual authentication between publishers, subscribers and message routers, enforces access control on message delivery and integrity control of message content. Access control is also a means to reduce traffic in a PubSub network since unauthorized message traffic will not be forwarded. A framework for providing identity management in a generic PubSub systems is presented and analyzed in this paper. The trust in the system relies to some extent on the use of hardware units for the protection of software integrity.

1 Introduction and PubSub security principles

The integration of a security model into a Publish-Subscribe distribution system is the objective of this paper. Its implementation employs services from an Identity Management System and the Trusted Platform Module (TPM) hardware unit.

In a *Publish-Subscribe* (PubSub) system, the information is not disseminated based on the addresses (or any individual property) of the receivers, but on *subscriptions*, through which the receivers specify their interest for information. Two forms of subscriptions are prevalent: One is to associate the information elements with a set of *topics* which are used as selection criteria in subscriptions, the other is to specify conditions for information of interest (like ``currency==USD'' or ``temp>90F''). In this paper the first form will be used since this is the most widespread approach.

The concept of *Identity Management* (IdM) introduces services for assured management of identities and their associated attributes: *Key pairs* for signing, encryption and authentication, *roles* for access control purposes [1] and other *attributes* that can support application services. For the remainder of this paper, the term *Attributes* also includes the concept of *Roles*.

Attributes may express the authorization of an entity and they should therefore be attested by a trusted party called an *Identity Provider* (IdP). The data structure used for that purpose is the *Identity Statement* (IS), where the identity, keys and attributes are stored and protected by the signature of the trusted IdP. The IS is somewhat similar to a Public Key Certificate, but an IS has a shorter life time and contains attributes as well as the public key. The short lifetime of an IS eliminates the need for a revocation arrangement.

The PubSub distribution principle has been demonstrated to be highly scalable. It allows route aggregation, multicast employment, loose coupling between parties, message queuing and sometimes message ordering guarantees.

1.1 Security concerns

On the other hand, the PubSub principle also poses a set of security problems which are often overlooked, like:

- Is the published information to be trusted as unmodified and authentic?
- Who is the publisher, and is it authorized to publish this information?
- Will the publications only reach authorized and authenticated subscribers?
- Who operates the message routers, and are the routers able to maintain privacy of subscriptions, and confidentiality and integrity of messages?

Protection of systems, including PubSub systems, is a multi-faceted problem. As Wang et al. [2] points out, it includes authentication, access control, information integrity, service integrity, confidentiality, accountability and availability. This paper focuses on how authentication and access control mechanisms provides information integrity and confidentiality, and how hardware-based units may support the need for service integrity.

The simple and widespread solution to authentication is to let every system keep its own registry of user names and passwords, an approach which is known to scale poorly and to create inconsistent and stale user information. A system based on centralized identity providers can offer a centralized registry and mutual authentication of client and service.

Access control in PubSub systems can be divided into those who enforce their policies either on connections, on subscriptions or on individual messages. The system being presented in this paper is the latter category, where each message being published is given an access policy which is checked for every subscriber before delivery.

Access control systems can be identity based, where privileges are assigned to individual users, or role based, where privileges are given to *roles* which again are assigned to users. Role Based Access Control [1] are known to scale better in terms of management resource requirements, but *role engineering* is necessary in order to avoid a combinatorial explosion of the role set. The use of RBAC access control in PubSub systems was investigated by Belokosztolszki et al. [3], but their work enforces the access policy only on subscription level. IBM's WebSphere MQ takes a similar approach, where access to "topics" (called *nodes*) are controlled by an ACL-based protection mechanism.

Our work extends, in some respects, the protection mechanism proposed by Fiege et al. [4], where the use of Attribute Certificates controls the reception of individual messages, although our work pays more attention to the problems related to identity management and how the access control mechanisms can mediate the message flows.

In the context of a PubSub distribution system, the use of Identity Management may increase the trust in the integrity and the confidentiality of the messages, and the control of who can post and subscribe to messages. This can be done in a scalable manner with loose coupling between the responsible parties:

- The integrity of messages is the concern of the subscriber, which should be able to express publication requirements and verify the correctness of the content.
- The confidentiality is the concern of the publisher, which should be able to specify the required authorizations for the subscriber to receive it.
- The authenticity of identities (routers and clients) is the responsibility of the Message Routers, which serves as the contact point to the transport infrastructure and requires that all entities authenticate themselves prior to operation.
- The integrity of identifiers, keys and attributes associated with an entity (subscriber or publisher) is the responsibility of the Authority, which operates an Identity Provider service for the purpose of issuing identity statements.

This "separation of concern" pattern reduces the necessary amount of coordination and control traffic, and offers a flexible arrangement for the expression and enforcement of the security requirements.

From an architectural perspective, access control adds an extra dimension to message routing in a PubSub system. There is a set of topics (hierarchical or orthogonal) associated with publications and subscriptions which mediate the message flow from publishers to subscribers. The additional requirement listed above can be distributed between routers so that messages do not travel further than necessary. There is no need to pass on a message if there are no authorized subscribers downstream (even if they subscribe to the given topics). Similarly, there is no need to pass on messages from publishers which are not approved by any downstream subscribers.

1.2 Integrity protection of routers

The descriptions in the previous sections indicate that the confidentiality of messages rely on the integrity of the message routers. If they fail to operate according to the subscription requirements in the messages, confidentiality may be compromised.

The approach taken in this paper is to employ hardware units called Trusted Platform Modules (TPM) to seal the configuration of the router nodes in order to detect malware etc. The TPM is non-bypassable and tamper-proof. The use of TPM will be discussed in Section 5.

1.3 Gismo IdM

The IdM prototype being presented in this paper was once developed with a service oriented, tactical environment in mind, with ample opportunities for cross domain operation and prudent networking protocols. In addition to the IdP services, it also offers a range of authentication protocols for use in different contexts. It is called *Gismo IdM* from a project called "General IT Security for Mobile Operations" at the Norwegian Defence Research Establishment.[5] Gismo IdM supports the PubSub environment excellently.

The contribution of this paper is a novel security model for PubSub service environment and its implementation, which partly relies on the services from an IdM, partly on a hardware based integrity protection. The integration of access control in the message routing decisions has not been observed in existing publications.

The remainder of the paper is organized as follows: Section 2 gives an architectural overview of the PubSub system which is being presented. In Section 3 the mechanisms related to publication selection and forwarding are described. Section 4 gives a detailed presentation of the underlying identity management systems (Gismo IdM). Furthermore, a presentation of how the TPM hardware module can cooperate with the IdM in order to support service integrity is discussed in Section 5. The paper presents its conclusions and suggestions for further research in Section 6.

2 Outline of the PubSub implementation

Figure 1 outlines the design of the proposed PubSub system. Observe the following properties:

- There are three types of actors: Publishers, Subscribers and Message Routers (simply called Routers for the remaining of the paper). A process can be both publisher and subscribers at the same time since they use the same interface.
- All actors belong to one *Community of Interest* (COI).
- The identities of the actors in a COI are governed by an *Identity Provider* (IdP) which issues *Identity Statements* upon request. The identity statements together with the (pre-loaded) private key enable the actor to authenticate itself when a connection is established.
- Messages (also called Publications) flow from one publisher to zero or more subscribers.

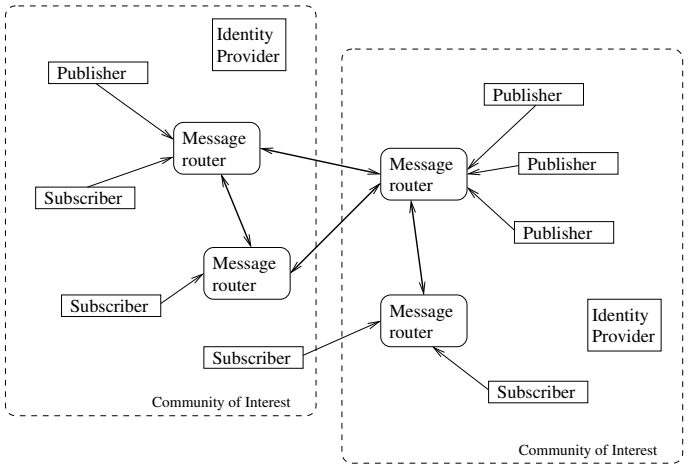

Fig. 1. PubSub system design outline

Figure 2 illustrates the representation of a subscription. It consists of a set of topic strings, which are to be interpreted in a hierarchical context: A subscription related to a

node in the topic tree also includes all sub-topics under that node. The subscription also includes a logical expression that must evaluate to "true" with respect to the publisher's identity attributes: The subscription shown in the figure requires that the publisher either has the attribute `nationality=''uk''` or `sec_clearance=''NATO secret''`. Otherwise, the subscription will not apply to messages posted by that publisher.

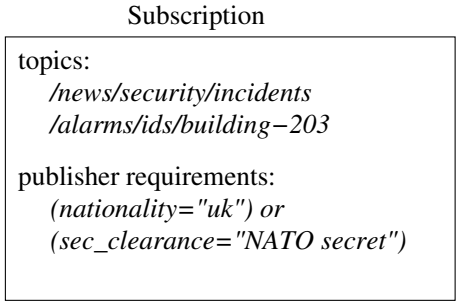

Fig. 2. Subscription representation

Publications are, in the same manner, annotated with a set of topic strings to describe its categories, and a boolean expression which must evaluate to "true" with respect to the attributes of the subscriber, called the *subscriber requirement*. Its function is to protect the confidentiality of the message, and it is the responsibility of the message routers to ensure that the confidentiality is not broken.

2.1 Functional requirements

As a basis for the design and analysis, the functional requirements to how an Identity Management system (IdM) will influence the properties of a PubSub system will now be described:

- The message flows from publishers to subscribers are mediated by topics of subscriptions and publications, as well as the identity attributes of the parties and the related attribute requirements. Messages will never be delivered to unauthorized subscribers or from unauthorized publishers.
- Client (publishers and subscribers) to router connections and router to router connections must be mutually authenticated.
- Cross domain authentication is possible where there are trust relations between the given IdPs.
- Publications are signed for integrity protection, and the identity statement of the publisher is included with the message.
- A subscription to one topic implies subscriptions to all sub-topics as well.

2.2 Non-functional requirements

To improve the scalability, resilience and reliability of the PubSub system, a set of non-functional requirements must be met:

- A router decision to stop a message on its way to the subscriber should be made as close to the publisher as possible, regardless if it is a matter of topics or authorizations.
- There should be no revocation mechanism for the identity statements. ISes should be issued with so short lifetime that revocation becomes unnecessary. A connection is terminated if the ISes involved in authentication expires without being renewed.
- There should be no need for configurations of the message routers except for loading their key pairs and enter the IP addresses of their link neighbors. No registration of client identities or topics need to take place.
- In the case where topics are hierarchically organized, the syntax representation of a topic need to list all ancestor topics as well (e.g. a full "pathname" from the top of the topic tree).

3 Operational details

The operation of the system will now be explained at the detail level of algorithms and protocol transactions. Some important data structures will also be described.

3.1 The subscription object

A subscriber expresses interest in posted publications based on topics and logical expressions of attribute values. Their syntax definition is as follows:

```
subscriptobj   = topics,expr
topics         = topic+
topic          = string
expr           = operand
                 | expr binary-operator expr
                 | unary-operator expr
operand        = literal | attribute
literal        = string | number
                 | true | false
attribute      = string
unaryoperator  = not | nameexists
binaryoperator = == | > | ismember | hastoken
                 | substring ...
```

The attribute values which enter into the logical expressions are taken from the identity statement of the publisher. This grammar is simplified for the purpose of discussion, e.g., logical operands cannot use the "$>$" operator etc.

3.2 Subscription matching

Let P denote the set of all publications. The subscription a matches a set A of publications, $A \subseteq P$. A publication p is a member of A if the boolean function $match(a,p)$ returns *true*, formally expressed in the following manner:

$$A = \{p | match(a,p) = true\}, p \in P \tag{1}$$

The selection made by the *match()* function maintains the *integrity* aspect of the delivery mechanism: In addition to the topicality selection, the publisher must meet the requirements set by the subscriber. These requirements are evaluated on the attributes of the publisher's identity statement, which are assigned and sealed by the Authority and distributed by the Identity Provider (IdP).

Please observe that a subscriber will not receive a publication if it doesn't meet the requirements set by the publisher. These requirements are expressed as a boolean expression, included in the publication object and evaluated on the attributes found in the subscriber's identity statement.

3.3 Merging of subscriptions

An important operation on subscription objects is the *merge* operation, during which a set of subscriptions are joined into one which represents the equivalent selection criteria. The operation is essential when a router wishes to announce the subscription of all its clients. The merging process takes two subscription instances (a,b) and produces an aggregate subscription c:

$$c = merge(a,b) \tag{2}$$

Where the set C matched by c relates to A and B so that

$$P \supseteq C \supseteq A \cup B \tag{3}$$

The `merge` function may create aggregate subscriptions which match superfluous publications. It is an assumption that the aggregate subscription can have a more compact representation if it is allowed a fraction of false positives, which we choose to call the *spill*. The spill set can be expressed by a set function in the following manner:

$$spill(a,b,c) = \{x | x \in C, x \notin (A \cup B)\} \tag{4}$$

which is simply the differential set $C \backslash (A \cup B)$. Since the spill represents a waste of communication bandwidth, the relation between the spill and the size of the aggregate subscription is interesting from an optimization perspective. It is reasonable to expect a size/complexity tradeoff of an aggregate subscription and the amount of spill, approximated by

$$|spill(a,b,c)| = \frac{k}{size(c)} \tag{5}$$

(k is simply a scaling factor). This tradeoff can be dynamically adjusted in the routers, since this is where the merging function is calculated, and this is where excessive publications are identified.

3.4 The publication object

A publication (sometimes simply called a message) is created by a publisher and sent to a router. The router will deliver it to locally connected subscribers and to neighboring routers in accordance with the service semantics and the existing subscriptions. The different aspects of publication and subscription routing will be described later. In this section, only the structure of the publication object will be described.

A publication object represents a publication as it is passed through the system. The elements of the object enables the system to enforce the confidentiality requirements of the message, and the integrity requirements set by the subscriber. The elements of the publication object are:

1. The actual message content (any serializable object)
2. The topics of the message
3. The identity statement of the publisher
4. A boolean expression which expresses the requirements which the subscriber must meet
5. An unique value used for loop detection
6. The publisher's signature

Through the inclusion of the publisher's identity statement and a signature, the subscriber can verify that the publisher meets the attribute requirements, and that the message is unmodified. The publisher is not allowed to include the identity statement, this is done by the router which remembers the identity statement from the authentication phase.

3.5 Routing tables

When a Message Router starts its operation, it will make connections to a number of other routers, based on its configuration. The population of routers thus forms a mesh networks, as shown in Figure 1, where there are indirect connections between some routers, and *routing information* becomes necessary.

The routes are constructed through the flooding of subscription objects, there are no separate routing messages in the system.

Every router will construct an aggregated subscription based on the subscriptions of all its connected clients, add a time stamp, and regularly pass it to all its router neighbors for the purpose of flooding. For the remainder of the paper, this structure is called the *AggSub* object.

Every router will keep a list of the AggSub objects it has received, and from which neighbor it came from. The time stamp of the object serves two purposes: To detect and discard duplicates during the flooding process, and to establish the best route based on the earliest AggSub object received.

A local timestamp is attached to every entry in this list, so that stale routes and disappeared routers will be removed from the list. The regular dissemination of AggSub objects serves as a leasing mechanism for liveness control.

The purpose of propagating not only the topics of interest, but also the attribute requirement is to discard unwanted/unauthorized publications as early as possible in its

forwarding path. The advantages of doing this is twofold: (1) To avoid wasting bandwidth and (2) to reduce the exposition of messages to message routers that are possibly compromised.

3.6 Publication routing

The distribution and use of AggSub-lists for routing information is simple and avoids separate mechanisms for route construction and subscription dissemination. It does not avoid route cycles, so a mechanism to detect and abort looping of publication messages must be in place.

When a router receives a publication, either from a connected publisher or from a neighbor router, it will consult the tables of connected subscribers and the AggSub list to determine who to forward it to. Locally connected clients are given the publications individually since they are separately connected.

Routers who are entitled to receive the publication are passed the publication through the neighbor for that route. Each neighbor are given at most one copy, and the neighbor from whom the publication was received is given none.

Loops are detected through the inclusion of a UUID in the publication object which are temporarily remembered in the router. Duplicates are silently discarded.

Client subscription table This table is built dynamically based on the connected clients. Each row represents a client and the columns contain the following information:

- An object representing the TCP connection
- The identity statement of the client
- The subscription objects of the client

During connection, the clients authenticate themselves through a protocol where the client demonstrates the possession of its private key associated with its identity statement. The client table is a trusted source of information about the clients and their subject attributes, a trust which is essential to the message passing mechanisms.

4 Identity Management details

The PubSub system being presented in this paper requires a trusted party which issues credentials and an infrastructure for the distribution of these credentials. The credentials are needed to authenticate a client or a router, and to attest the attributes associated with the subject for access control purposes. The message forwarding mechanisms described earlier in the paper is considered to be access control operations, since they support the requirements of message integrity and confidentiality.

Systems that support these operations are commonly known as *Identity Management* (IdM) systems. At one end of the IdM system range we find Single Sign On (SSO) systems with client-authentication only and strong coupling between the parties. At the other end of the range there are web-service oriented system which require validation

of credentials in both ends of a connection, with the ability to operate in a cross domain environment with loose coupling between the management domains.

For the PubSub system being described in this paper, the existing *Gismo IdM* experimental IdM system was used as a trusted third party.[5]

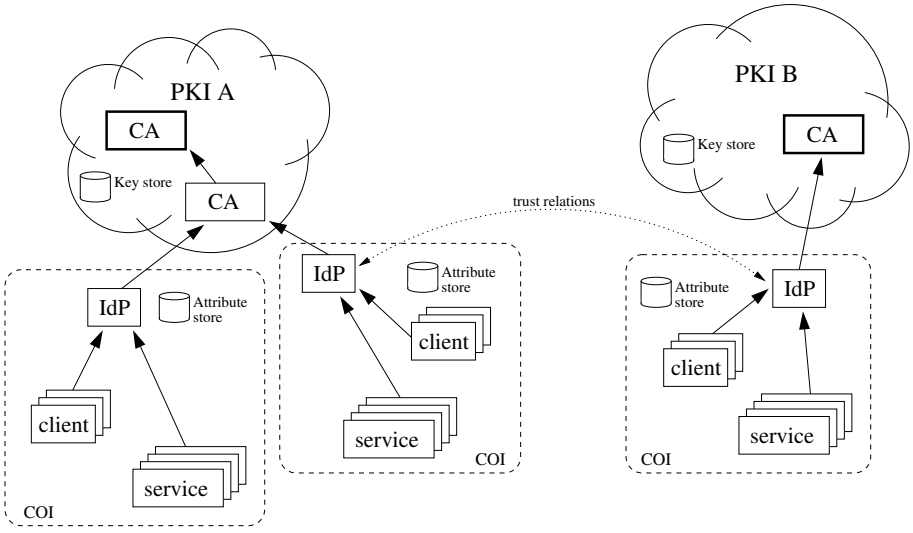

Fig. 3. The functional components of the Gismo IdM. Observe that the IdP serves one single COI, and the trust relations are formed between COIs, not domains. Key management is handled by the PKI whereas the attribute management is done by the IdPs on the COI level.

A architectural view of the Gismo IdM can be seen in Figure 3. Its notable properties are:

- Identity statements contain subject identifier, subject attributes, subject's public key, issuer identifier, issuer's public key and an expiration time.
- Identity statements are issued with a short lifetime. No revocation arrangement is therefore needed.
- Identity statements are issued by an *Identity Provider* to subjects that are properly registered in, e.g., a PKI. The subject's private key and the IdP's public key must be pre-installed in the subject's computer for the identity statement to be useful.
- *Guest identity statements* can be issued to subjects who can present an identity statement issued by an IdP to which there exists a *trust relationship*. Guest identity statements allow parties belonging to different management domains to authenticate themselves.

For the use of the Gismo IdM in a service oriented (SOA) environment in a military disadvantaged network some efficient authentication protocols were developed. They minimize the number of protocol round trips by piggybacking authentication data on

the actual service request and service response, and they minimize the required state space in the server through nontraditional replay protection mechanisms.[6]

One of these authentication protocols has been used in the PubSub system as a means for mutual authentication of router-to-router and router-to-client connection. Consequently, all clients and routers need to be registered with an IdM instance and have their key pair installed prior to connecting to a router.

4.1 Cross domain connections

In those cases where the parties of a connection belong to different domains, i.e., they trust different IdPs, they cannot directly validate each other's identity statements. In order to authenticate across IdM domains, *guest identity statements* were introduced. Figure 4 shows how a client X_a from IdM domain a presents its identity statement, termed $(Id_x)_a$, to the IdP of domain b. On the condition that there exists a trust relationship between the domains a and b, IdP_b can issue a guest identity statement with the same content and its own signature. The guest IS is termed $(Id_x)_b$ and may be validated by any member of IdM domain b. For the opposite direction, authentication will require the identity statement IdP_a issued by IdP_b in order to construct a signature chain back to the trust anchor of a. This IS is termed $(Id_a)_b$ and is enclosed in the guest IS response.

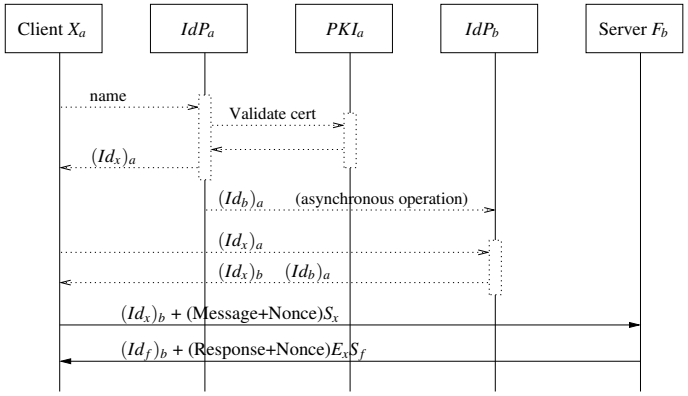

Fig. 4. The authentication protocol for the PubSub service. There is no replay attack protection since they are not considered as threats, but the response need to be protected for reasons of response replay and information compromise.

Figure 4 also shows the details of the authentication protocol. The design choice has been to sign the request for integrity protection, and to encrypt the response as a means of replay attack protection. The response also contains the necessary information to authenticate the responder.

This protocol was developed as a service invocation protocol in [6], and has been used as a connection protocol in the PubSub system. The symmetric key which is used

in the encryption of the response (as a part of the asymmetric encryption operation) is subsequently used for the protection of the message traffic.

4.2 Protection of message integrity

In addition to the control of the delivery mechanisms, the IdM-supplied credentials also offer the protection of publication integrity. As pointed out in Section 3.4 the publication object includes the identity statement of the publisher and a signature which seals the content of the message.

The receiver can check the integrity of the message and the authenticity of the publisher by verifying the signature and validating the identity statement.

Validation requires trust in the IdP of the publisher, which exist if they belong to the same IdM domain or to domains with a trust relationship. If, however, they belong to domains between which there exist a *chain* of trust relations, then the validation cannot take place. Trust relations are per definition *lateral*, not transitive.

4.3 Renewal of identity statements

In a SOA environment where services are short lived, i.e., they finish very soon after being invoked, it suffices to check authenticity and authorizations during invocations. In a long lived service environment, where the services are active for hours after invocation, the credentials should be checked for expiration during this period.

The PubSub system is a long lived service, where the connection to a router may be held active for hours and days. Since the credentials used during connection establishment have an expiration time shorter than that (typically a small number of hours), their validity should be monitored so *no connection can be active if the credentials are expired or invalidated*.

For the connection to be maintained across credential expiration periods, the identity statement must be *renewed* during a connection period. The owner of an identity statement must contact the IdPs before expiration to get a new issue and send it to all its connected neighbors (clients and routers). Re-authentication is not necessary.

The recipient of the identity statement should validate it and check that the subject attributes meet the connection requirements. It should also replace the old set of attributes so that new subscriptions and publications reflect this set.

Existing subscriptions may be incorrect with respect to the new attribute set. Subject attributes are not included in the subscriptions when propagated through the network (only topic information and publisher requirements are). Therefore, the flow of publications will not need to be altered due to the new identity statement. The new set of subject attributes will always be present in the router which connects the client and make sure that the publications are delivered to clients based on their most recent attribute set.

5 Trusted binding of Message router operation

The confidentiality of publications is the responsibility of the message routers, which means that they must be trusted to operate correctly and not leak information to unauthorized subscribers.

Even if a message router has correctly authenticated itself to its neighbors, one cannot exclude the possibility for a message router to be infected by malware, operate incorrectly or to have been compromised in some other way. To some extent, it might possible to strengthen the trust between nodes by resorting to a *trusted binding* between a node identity and its own hardware and software configuration.

The concept of trusted binding has been introduced by Hegland et al. [7] as a way to reduce the burden of authentication in a network with low resources, and is tightly connected to the idea of transitive trust. If an entity A has authenticated and trusts an entity B, and B has a trusted binding with C, then A will also trust C without further need for authentication. In a way, this is what already happens in our framework: a message router A trusts another message router C because it presents a proof of authentication (the identity statement) issued by an IdP_b, which A trusts. However, we want to extend this concept to be able to trust, besides the identity of a message router, also its *integrity*. We accomplish this by integrating a TPM based attestation component in our identity management framework.

A TPM (Trusted Platform Module) [8], is a cryptoprocessor that enables a machine to perform, among other things, a *trusted boot* and thereafter to report in a trusted way (*attest*) its running software and hardware configuration to a third party. The trusted boot measures all software components being run on the machine (by computing the hash of their binaries) from the boot moment until the OS takes control, and stores these measurements in the TPM registers (PCRs) in a tamperproof manner. The measurements can then be signed with the TPM private key (which never leaves the TPM), and sent to a third party which can verify them against a database of approved configurations. The procedure is illustrated in Figure 5. This can allow an IdP to verify not only the identity of a message router, but also that it was booted with genuine and updated software. This extra information can then be added to the identity statement as a new subject attribute: `{TMPintegrity=AssuranceLevel2}`. Neighboring nodes may require this assurance in order to accept the nodes' participation in the router network and, possibly, to create clusters of routers with different assurance levels.

Similar work has been done with IdM systems based on Kerberos [9] and OpenID [10], although the main goal is to improve the user experience by eliminating the need for passwords.

One limitation of the trusted boot approach is that the trust in the system is built by the boot process and passed transitively to the OS, which takes over the system once the boot is finished. This means that after this point we should trust the OS to measure correctly itself and the processes it is running in order to perform a trusted attestation. Unfortunately, OSes are often quite large and complex and are likely to contain vulnerabilities that can be exploited by an attacker. Obviously, an infected machine cannot be trusted to report correct information about itself to a third party. Besides, if a system is compromised, the machine must be rebooted again in order to establish a new chain of trust. This is why the trusted boot is said to be based on a *static root of trust*.

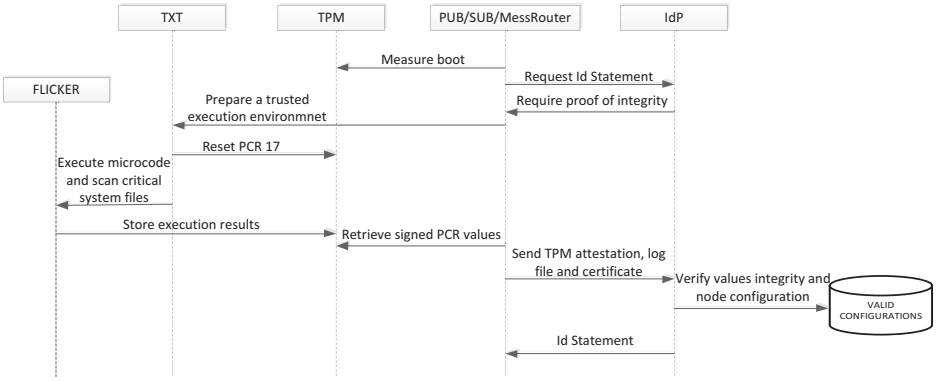

Fig. 5. This diagram shows an example of attestation process (at very high level) that leverages on both the TPM, a classical *trusted boot* process, and a TXT enabled Intel processor which can be used to execute Flicker microapplications in a trusted setting. In this case the microapplication is a virus/malware scanner that is run securely before sending the node integrity attestation to the Identity Provider. The result of the scan is also sent as part of the TPM attestation.

Although continuous monitoring and measuring of all the system in search for infections or malfunctions is unrealistic, research is ongoing to define new techniques that can help attesting the status of a running machine. Property based attestation is one example [11]. A better alternative, however, might consist in using a *dynamic root of trust*. The recent Intel and AMD processors offer a special instruction that allows the system to create a secure and trusted hardware environment for the execution of some given code, even if the rest of running system is compromised [12]. Therefore, trust in the system can be established *dynamically* at any time, independent of the boot process. The TPM is also involved in this process so that a node can attest that a certain piece of code was executed securely.

In particular, a framework called Flicker [13] allows to write micro applications that can be run safely at any time even the current OS is compromised. A report about the code execution can also be generated and certified with the help of the TPM in a fashion similar to the attestation process.

Associated with the PubSub research, a study is ongoing to understand how a dynamic root of trust based application like Flicker can be integrated with our framework. Our aim is to design an identity management system where authentication and valid identity statements can be issued also if some nodes in the network are compromised, and would normally not be trusted. The remaining probability for incorrect operation may be subject to calculation on forwarding path length: The higher number of routers used in a forwarding path for information, the higher is the aggregated probability for incorrect operation. These calculations can also consider the strength of crypto algorithms in use, the number of COI domains involved etc.

6 Conclusion and future research

This paper has described a novel model for protecting integrity and confidentiality of messages in a PubSub system. Pivotal to the model are the services of an IdM, which supports the authentication and access control operations, and the integrity protection offered by a non-bypassable hardware unit called TPM.

A proof of concept prototype has been written in Java to study the functional aspects of the proposed PubSub principles. A deeper evalution in the context of a military information system is planned in the near future. Also, the protection mechanisms that rely on a TPM hardware unit is being developed for the time being as an ongoing research project.

References

1. Sandhu, R., Ferraiolo, D., Kuhn, R.: The NIST model for role-based access control: towards a unified standard. In: RBAC '00: Proceedings of the fifth ACM workshop on Role-based access control, New York, NY, USA, ACM (2000) 47–63
2. Wang, C., Carzaniga, A., Evans, D., Wolf, A.L., Wolf, E.L.: Security issues and requirements for internet-scale publish-subscribe systems. In: 35th Hawaii International Conference on System Sciences (HICSS-35), Big Island. (2002)
3. Belokosztolszki, A., Eyers, D.M., Pietzuch, P.R., Bacon, J., Moody, K.: Role-based access control for publish/subscribe middleware architectures. In: Proceedings of the 2nd international workshop on Distributed event-based systems. DEBS '03, New York, NY, USA, ACM (2003) 1–8
4. Fiege, L., Zeidler, A., Buchmann, A., Kilian-Kehr, R., Mühl, G., Darmstadt, T.: Security aspects in publish/subscribe systems. In: Third Intl. Workshop on Distributed Event-based Systems (DEBS04, IEEE (2004)
5. Fongen, A.: Architecture patterns for a ubiquitous identity management system. In: ICONS 2011, Saint Maartens, IARIA (Jan. 2011)
6. Fongen, A.: Identity management without revocation. In: SECURWARE 2010, Mestre, Italy, IARIA (July 2010)
7. Hegland, A.M., Winjum, E., Hedenstad, O.E.: A framework for authentication in nbd tactical ad hoc networks. IEEE Communications Magazine **49**(10) (2011) 64–71
8. Trusted Computing Group: TPM Main Specification. http://www.trustedcomputinggroup.org/resources/tpm_main_specification Online, Accessed Mars 2012.
9. Leicher, A., Kuntze, N., Schmidt, A.U.: Implementation of a trusted ticket system. In Gritzalis, D., Lopez, J., eds.: SEC. Volume 297 of IFIP., Springer (2009) 152–163
10. Leicher, A., Schmidt, A., Shah, Y., Cha, I.: Trusted Computing enhanced OpenID. In: Internet Technology and Secured Transactions (ICITST), 2010 International Conference for. (nov. 2010) 1 –8
11. Nagarajan, A., Varadharajan, V., Hitchens, M., Gallery, E.: Property based attestation and trusted computing: Analysis and challenges. In: Proceedings of the 2009 Third International Conference on Network and System Security. NSS '09, Washington, DC, USA, IEEE Computer Society (2009) 278–285
12. Grawrock, D.: The Intel Safer Computing Initiative: Building Blocks for Trusted Computing. Engineer to Engineer Series. Intel Press (2006)

13. McCun, J.M.: Reducing the Trusted Computing Base for Applications on Commodity Systems. PhD thesis, School of Electrical and Computer Engineering, Carnegie Mellon University (2009)