Institute of Parallel and Distributed Systems
University of Stuttgart
Universitatsstrasse 38
D–70569 Stuttgart

Diplomarbeit Nr. 3307

Confidential subscription clustering in a publish/subscribe system

Manuel Bischof

Course of Study:	Computer Science
Examiner:	Prof. Dr.rer.nat. Dr.h.c. Kurt Rothermel
Supervisor:	MSc. Muhammad Adnan Tariq
Commenced:	05.03.2012
Completed:	04.09.2012
CR-Classification:	C.2.1, C.2.4, K.6.5, D.4.6

Abstract

Broker-less content-based publish/subscribe systems offer the chance for flexible and loosely coupled many-to-many communication. Initially, no centralized component is needed. Routing is done by the peers themselves by looking at the content of the messages. This works fine, as long as no sensitive data is published. In this case, users may want to encrypt their data and even keep their subscriptions confidential.

At first glance, this contradicts the content-based routing paradigm. The work of Tariq et al.[28] presented the first system, that supports both by introducing a weaker notion of subscription confidentiality. Identity-based encryption is adapted to suite the requirements in such a system.

This thesis will analyze the approach, point out some still existing problems and try to improve them. The current system uses one-hop flooding during event dissemination, which yields a high number of false positives. Simulations will show, that confidentiality can be kept nearly as high with much less false positives. Another issue is private key management as the number of keys to maintain is in order of $O(\Sigma_{i=1}^{d} \log_2 T_i)$ with d attributes in total and $T_i = \frac{UpperValueLimit_i - LowerValueLimit_i}{Granularity(i)}$, where Granularity(i) determines the finest step value of attribute A_i. As the number of total attributes in a real system may be much greater then the attributes needed by an event, there are two problems: 1) how to specify the attributes that are not required by the event and 2) the cost of cryptographic operations is dependent on the total number of attributes.

In the second part of the thesis we will provide another approach that decouples events and subscriptions from the total number of attributes. Timings, based on an implementation using the Pairing-Based Cryptography library (PBC)[10] will be given.

Contents

List of Figures

1 Introduction

The established client-server model is well suited for many applications, where one known server is in contact with only one user per session. Even if this simple scenario is extended to 1-to-m communication, still sender and receiver need to know each other. Many modern applications require m-to-n communication, e.g. to allow users to interact with each other. If all of the messages need to be sent over a server, the system may be not scalable. Moreover, client-server interaction is initiated by the client. This is not suitable for all scenarios, as there exist many event-driven applications, like stock quote services. Each user, who wants to be notified as soon as a stock quote rises over a specific value would have to poll the server individually every now and then, even if no new information is present. If interaction could commence by the server, only one message would suffice, because it knows when to send it.

This is why *publish / subscribe* systems become increasingly popular. Here, communication is initiated by the sender. Users can still have two roles: 1) Publishers, who write messages (often also called events or notifications) without declaring any special receiver and 2) subscribers, who announce their interests in the form of filters and will receive the events. In most systems a fixed broker infrastructure, the close match of servers, handle the part in between. It uses subscriptions to route published events to all subscribers with matching filters. By this means, sender and receiver are decoupled, as they have no information about each other and do not need to be logged into the system at the same time (if brokers store the events). Most frequently used is the most expressive *content-based* routing model[3, 25, 24], that decides only on basis of the actual message content, where to forward messages. Often, these brokers are trusted, so that addressing confidentiality is quite easy.

Recent work in the field of publish / subscribe is frequently based on the idea of *peer-to-peer*, resulting in *broker-less* systems. The tasks of the brokers needs to be delegated among the peers themselves and the infrastructure has to be self-organizing. As every peer is possibly part of the routing overlay and there cannot be any trusted routing infrastructure, confidentiality is much harder to achieve. Not only confidentiality, but also Quality of Service and clustering are discussed[26, 27, 29, 30].

1.1 Provision of security mechanisms

As the awareness of privacy and confidentiality grows steadily and it is long since it reached the world wide web, also publish / subscribe systems are concerned. Three types of security are important: First, authentication makes sure, that users are allowed to send or receive messages. In some systems, users will have to pay a fee or need to register first.

Subscription confidentiality assures, that the interests of subscribers are kept private. In a broker-based environment, where brokers are trusted, the solution is easy: Peers only need to tell their subscriptions to the broker and they can use this information to match them against incoming events. Without brokers, strong subscription confidentiality cannot be kept, unless broadcast is used, which yields a lot of false positives and is not scalable for obvious reasons.

Publication confidentiality on the other hand pays attention to the events. Their content should be kept private, so that only authorized subscribers can read them and nobody can modify the message. Encrypting and signing would solve this problem, but this is a contradiction to the content-based routing paradigm or the expressiveness needs to be limited. Moreover classic end-to-end encryption needs the two parties to share keys, no matter if symmetric or asymmetric encryption is used. This again conflicts with the loose coupling reached by publish / subscribe.

Pietzuch (Hermes[16]), Pesonen et al.[15], Opyrchal[14] all address security in publish / subscribe systems, but all rely on brokers. But Tariq et al. is the first to propose a system[28], where all, authentication, subscription and publication confidentiality are handled in a broker-less content-based publish / subscribe system. A weaker notion of subscription confidentiality is introduced to ensure, that peers cannot infer more than a containment relationship about the subscriptions. Peers are then clustered in hierarchical ordered trees to build an event dissemination infrastructure. Identity-based signcryption[33], is used for authentication by encrypting and signing the messages. On the other hand, only authorized subscribers are able to decrypt and verify the events. Indeed a key server is used for this method, but scalability is preserved, as it only needs to setup the initial parameters and, in contrast to public-key infrastructure (PKI) systems, store only one private key, independent of the number of users. Each users only needs to ask the key server for its private key once per epoch. This load can be easily balanced by replicating the key server.

Whereas this is seems a promising solution regarding confidentiality, other drawbacks arise. As peers do not know subscriptions of other peers, one-hop flooding is used to disseminate events to children in the trees, but not all of the children will have a matching subscription. So a lot of false positives are generated. The second issue is key management, as the keys to maintain for each peer is $O(\Sigma_{i=1}^{d} \log_2 T_i)$ with d attributes in total and $T_i = \frac{UpperValueLimit_i - LowerValueLimit_i}{Granularity(i)}$, where Granular-

ity(i) determines the the minimal value difference of attribute A_i. In general, the total nubmer of attributes d may be much larger than the set of relevant attributes d' for one event.

Unfortunately, no evaluations of the confidentiality were made so far, but this thesis provides an analysis, using a simulation written in java with PeerSim[12] and aims at finding weaknesses in the concept and provides a suggestion to improve them.

1.2 Organisation of this thesis

The remainder of the thesis is structured as follows:

- next, chapter 2 gives an overview over related work.
- Chapter 3 explains the underlying paper with techniques used and states some of its problems.
- Chapter 4 will analyze the current approach with the help of a simulation. Also a variation of the event dissemination protocol is discussed.
- In chapter 5 identity-based signcryption and attribute-based encryption will be combined to create an alternative cryptographic scheme that allows the use of an arbitrary number of advertisements and subscriptions.
- At last, chapter 6 will conclude this thesis and give some hints about future work.

2 Related Work

2.1 General overview

In recent years a lot of research effort has been spent on building scalable and expressive content-based publish / subscribe systems[3, 13, 25] or confidentiality in form of encryption and authentication[6, 21, 32, 5, 23, 9]. However the combination security issues in broker-less systems is rarely addressed.

Another paper[8], that was released at about the same time as [28] also focused confidentiality in publish / subscribe networks, but with brokers. Their idea is to encrypt events with attribute-based encryption and handle subscription confidentiality by sending filters encrypted to the brokers and then use mechanisms of encrypted search on them.

[1] gives an good introduction into identity-based encryption in general, an asymmetric cryptosystem type, that uses obvious but unique identifiers like names or addresses as public keys. Then two systems are introduced: Cocks' method uses quadratic residues and the more famous approach of Boneh-Franklin, using bilinear maps. More about the latter one can also be read in [11].

Based on IBE exists an interesting paper[33] about identity-based signcryption. Here, encryption and signature are combined in one step and its security is proven secure without using random oracles. This technique is useful, when authentication and confidentiality are both required.

Attribute-based encryption (ABE) was first presented by Sahai and Waters[19]. Enhancements were made[18, 7] to support ciphertext-policy attribute-based encryption (CP-ABE) and key-policy attribute-based encryption (KP-ABE). The intention is to drop the need to encrypt with one key for a particular user, but to be able to specify an access structure, that determines, who can read the message. In CP-ABE the access structure is embedded in the ciphertext, meaning, the sender can specify, who shall be able to decrypt the message. On the other hand, KP-ABE shifts the responsibility to the receivers. The other private key is in each case labeled with attributes, which can be matched against the access tree. [19] made it possible to decrypt a message, as soon as a certain threshold of similarity was exceeded. This is useful for biometric access control, as measurements are faulty.

The last two mentioned methods, ABE and signcryption, were used in the original approach[28] as well as in this thesis to construct a cryptosystem, that is able to encrypt and sign at the same time as well as to handle access control.

At last, I want to mention [17], that sharpens the awareness to traffic analysis, as

not only breaking encryption, but already knowing message flows can reveal some information. Assume a state, that can track traffic from an terrorist website back to the users. Not all of them may be criminals, but at least suspicious. One hint is already given by an intersection attack, that tries to map users, that are online at the same time, if there is a bidirectional communication between them. Timing Attacks look at the round trip times for known route lengths. We assume in this paper, that an attacker may track messages through our network as long as it has knowledge of the links.

2.2 p^5: A protocol for scalable anonymous communication

2.2.1 Basic idea

p^5[23] (Peer-to-Peer Personal Privacy Protocol) is based on a naive solution of broadcasting in a peer-to-peer environment. If all peers participating in the system send dummy messages with a fixed length with a fixed time between two packets, an adversary could not gather any information by observing network traffic, as all peers behave in the same way. The dummies, also called noise, are replaced with chunks of the real message for communication. Each packet will be encrypted with a public key of the receiver to protect the content. It is clear, that this method cannot be scalable as there exists a trade-off between limiting sending rates and overhead for noise. Receiver anonymity is granted, as the receiver could be anywhere in the broadcast group. To provide sender anonymity a mechanism to hide the sender address is needed. This can be achieved by implementing the broadcast as a ring. By this means, no peer can determine if it is the first receiver of this message.

2.2.2 Reaching scalability

p^5 achieves scalability by introducing a hierarchy of broadcast channelsΛ. Λ is a complete binary tree where each node is a broadcast group, represented by a bit string. Users are mapped to one of these groups by hashing their public key and choosing an anonymity coefficient m. According to these two information, the user is assigned to the group, that matches the first m bits of its hashed public key. The smaller m, the higher up in the tree is the user placed and less anonymity can be provided. But on the other hand, communication is more efficient.

Anonymity is based on indistinguishability from other users and thus depends on the size of the broadcast groups and the number of groups a user could be part of.

2.2.3 Adapting this protocol

p^5 is designed to work for one-to-one communication as a symmetric encryption is used for message confidentiality. For the use in a publish / subscribe system there are neither methods to keep subscriptions confidential nor to map users to a group of receivers. This renders the approach useless for our purpose.

2.3 Information slicing: Anonymity using unreliable overlays

Information slicing[9] proposes an innovative method to provide anonymity without any encryption. Any information is split into d pieces and sent along different routes, so that no user receives more than one piece of information.

2.3.1 Premise

Information slicing assumes, that a user has access to d distinct IP addresses and Internet access points, one of which needs to be uncompromised. The latter is denoted as source S, the others as pseudo-sources S'. S could be a home computer, whereas S' can also be in school, work or even an Internet cafe.

2.3.2 Message scrambling

Sometimes even a partial message can already reveal important information. In a war scenario a message "Attack at dawn" can be split up into d = 2 pieces with m_1 = "Attack at" and m_2 = "dawn". If m_1 would be intercepted, the enemy could come to the conclusion, that he will be attacked, even if he does not know when.

To prevent this, the message is multiplied with a random $d \times d$ matrix A, that needs to be invertible.

$$\vec{I^*} = \begin{pmatrix} A_1 \\ \vdots \\ A_d \end{pmatrix} \vec{m} = A\vec{m}$$

The sender will then send I_i^* and A_i to the receiver on d distinct paths. I_i^* denotes the i-th line of $\vec{I^*}$ and A_i the i-th line of matrix A respectively.

The original message can be restored by the computation of

$$\vec{m} = A^{-1}\vec{I^*}$$

2.3.3 Routing

The remaining question is how to find the d distinct paths. This is done by choosing a path length L and arranging disjoint random nodes in L stages. The receiver will

be on any of these. For a split factor d=2 and L = 3, let's assume path 1 with nodes in the following order S, U_1, U_2, U_3 and path 2 with nodes S', L_1, D, U_3, where S and S' are the source and pseudo-source and D is the destination. To establish the paths, the nodes will be informed about their next hops. Each node, beginning with the sources as stage 0, will send one message to each of the d nodes in the next stage containing the scrambled routing information of all the next hops on the path. After each stage, a random piece of information needs to be included to keep the message length constant.

2.3.4 Adapting this protocol

Unfortunately, there is no reasonable way to adjust this protocol to work in publish / subscribe systems as it would be great to abandon encryption, as it always needs a lot of computation time and thus resources. But the design is not designed for multiple receivers, as the path length would increase dramatically. Also the need to publish information from multiple sources in parallel is infeasible for real-time applications

3 Background and problem description

3.1 Identity-based encryption

In a large publish/subscribe system it is infeasible to maintain one public/private key pair per user, as it is needed in Public-Key Infrastructures (PKIs). The objective of Identity-based encryption (IBE) was to minimize the need for Certification Authorities. IBE was first introduced by Shamir[22] in 1985. Since then, it was developed further[19, 7, 18]. The idea behind IBE is to use any uniquely identifying string, like an email-address or a name as public key for each user. Most approaches[2] use a *bilinear map* which is a pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_\mathbb{T}$ for cyclic groups of order p with the following properties:

1. Bilinearity: $e(g_1^a, g_2^b) = e(g_1^b, g_2^a) = e(x, y)^{a*b}$ for all $g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2$, and $a, b \in \mathbb{Z}_p$.
2. Non-degeneracy: $e(g_1, g_2) \neq 1$, for all $g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2$.
3. Computability: e can be efficiently computed.

e is called symmetric, if $\mathbb{G}_1 = \mathbb{G}_2$. For cryptographic use, p is some large prime and the groups are often elliptic curves[11]. An alternative would be using quadratic residues[4, 1].

As described in [1], a generic IBE scheme consists of four steps:

1. **Setup:** Initialize the groups, bilinear map, hash functions and other public and private parameter
2. **Extract:** Return a private key for a given ID and the public parameter to the user
3. **Encrypt:** Generate a ciphertext for a given message and the ID of the receiver
4. **Decrypt:** Uses the private key to decrypt a ciphertext and restore the original message.

Figure 3.1 shows, how a message can be sent and received in IBE. The message can already be encrypted, as soon as the public parameters are published, with the receivers known identity. The receiver can then ask a key server for his private key by authenticating. This private key can be used for any future messages.

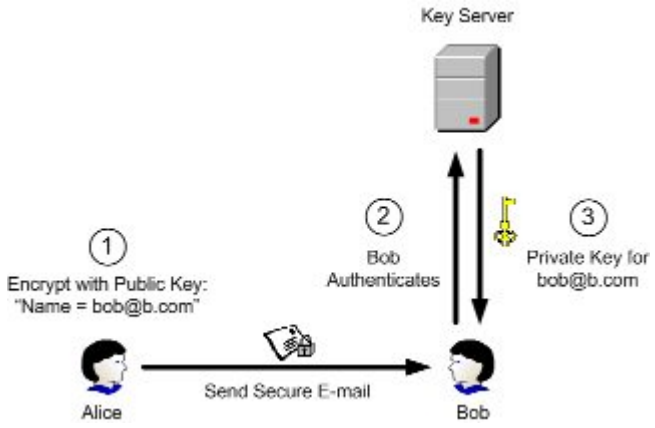

Figure 3.1: Scheme of identity-based encryption[1]

Note, that the key server only stores one private key. Users need their private keys and the public parameters to be able to communicate with every other user in the system. For this reason, IBE is suitable for the use in a large publish/subscribe system in contrast to common PKI systems. Certificates are no longer needed, as users are bound inherently to their identity. Instead of Certificate Authorities, a Private-Key Generator is used for setup and key distribution.

3.2 Providing security mechanisms in broker-less content-based publish / subscribe systems

The basis for this thesis is the work of Tariq et al.[28]. In contrast to most other papers in this area, security and confidentiality will be achieved in a *broker-less* publish/subscribe system. Roughly said, this works as follows: The event space is divided for each attribute separately and each attribute is assigned an own tree. Peers are added in a hierarchical manner in the trees according to their subscriptions. Coarse subscriptions are near the root, finer ones lower down to the leaves. Thus the peers know, they have a coarser or equal subscriptions then their children. Therefore we need a *weaker subscription confidentiality*. As routing-strategy each peer needs to check, if an received event matches its subscription. If so, the event needs to be send to parent and children, else only to the parent. A combination of identity-based encryption (IBE)[7] and signcryption[33] is used to keep events and subscriptions confidential.

I will describe this approach in detail now and then point out some issues afterwards.

3.2.1 System model

Content-based publish/subscribe

This approach uses the content-based publish / subscribe model without any brokers. We denote Ω as the set of all d distinct attributes and call it *event space*. Each attribute A_i has an unique *name* and a *domain*, the range of possible values, of any ordered data type like strings or numeric values.

Subscriptions are modeled as filters, which consists of a conjunction of d predicates P_i, each containing the attribute A_i, a compare operator for the data type of A_i and a value.

Events are denoted similar to subscriptions as d tuples of attribute and value, the difference is that exact values are given, so that they can be *matched* against subscriptions. A subscription f_1 is said to cover or contain another subscription f_2 if $f_1 \supseteq f_2$, that means f_1 matches the all of the events of f_2 and can match more.

Attacker model

We focus on the main part of Tariq's work, thus we assume, there is a secure way for communication with the key server and key distribution. This can be for instance any known transport layer security like SSL.

Next, we suppose that peers do not trust each other and everyone could be malicious. Nevertheless authorized peers behave as intended. Publishers only send valid events and subscribers do not share their information about other subscriptions and the content of messages. Last case would be equal to solve the digital copyright problem. In the case of malicious publishers, spam with faked events is supposed. Malfeasant peers in general try to get more information about the subscriptions of other users then the weaker subscription confidentiality allows to and subscribers try to read events, they are not authorized to decrypt. Still, malicious peers will stick to the protocol and forward messages as any regular user.

A passive adversary with global view completely outside the system is also considered, however there can be communication links, that it cannot observe. It will eavesdrop on the accessible links and gather information provided by malicious nodes, thus we assume all malicious nodes are colluding. This data is analyzed to break our security assumptions.

During event dissemination, we expect that an attacker can distinguish multiple events and thus can handle each notification separately. The physical address is visible to everyone and different subscriptions on the same peer can be told apart e.g. by a differing port. So each attribute tree will be handled individually.

3.2.2 Event space decomposition

To be able to cluster the peers, we need to divide the event space in distinct parts, so that containment relationships can be derived. For decomposition we introduce *dz-expressions*, to identify subspaces. dz-expressions are binary strings or the symbol ϵ to denote the whole event space.

Numeric attributes

For numeric attributes, the range can always be bisected and labeled with '0' for the lower half and '1' for the upper half. Assume a weather service, which publishes information about the temperature. The initial range is $\epsilon = [0, 100]$ (in %). The first bisection would yield $dz_0 =$ "0" $= [0, 50]$ and $dz_1 =$ "1" $= (50, 100]$ the second one $dz_2 =$ "00" $= [0, 25]$, $dz_3 =$ "01" $= (25, 50]$, $dz_4 =$ "10" $= (50, 75]$ and $dz_5 =$ "11" $= (75, 100]$. Thus, the number of bisections correlate with the length of the dz-expressions and indicate the granularity. Subscriptions and advertisements are ranges or hyperrectangles (see the case in the next paragraph), whereas events are single points.

We use this decomposition for each attribute tree separately for effectiveness, although it is possible to apply this mechanism for multiple trees at once. For that, an event space with d attributes is seen as a d-dimensional space, where each dimension represents an attribute. Thus the terms "attribute" and "dimension" are sometimes exchanged. Now the first bisection is meant for the first dimension, the second for the next and so on. The above example is extended with another attribute "humidity", for simplicity also in the range [0, 100]. The decomposition can be seen in figure 3.2. The example subscription f_1 would yield the dz-expression {01} and f_2 would need the two expressions {000, 010}.

Using this method, containment relationships can be determined easily by comparing the dz-expressions. If dz_0 ("0") is a prefix of dz_2 ("00") then dz_0 is coarser than dz_2, meaning $dz_0 \supseteq dz_2$.

String attributes

The above mentioned technique works intuitively for numeric attributes, but can be used with any ordered data type with limited bounds. Strings could be hashed or given a maximum length. Subsequently, prefix matching can be applied to divide the domain. Better are *tries*. In a trie, nodes are always prefixes and edges are labeled with characters. They are ordered in a manner, that each node is the prefix of its children.

For subscriptions, any node can be chosen, whereas events are always leaves. As tries are optimized for fast string look-up and prefix matching containment relationships and event matches can be easily calculated.

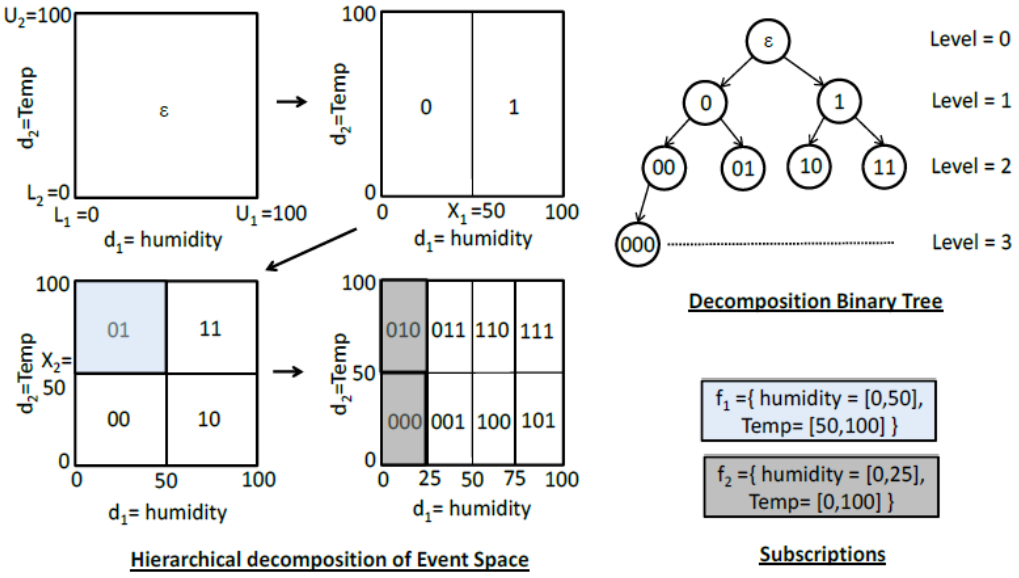

Figure 3.2: Event space decomposition for two numeric attributes[28]

3.2.3 Credentials

Credentials need to be provided by peers to the key server for authentication. They consist of capabilities and the proof of identity. The second part can be an arbitrary identifying method like a password, challenge-response or hardware supported techniques. As we are not interested in authentication techniques right now, we will focus on the capabilities, which are given by dz-expressions.

How keys are related

Now that we know how to divide the event space into subspaces and create dz-values, we can discuss the role of the key server, which is mainly to authorize credentials and to generate private keys.

The key server is only needed in steps 1 and 2 of the generic IBE scheme, as described in 3.1. The first thing to do in a fresh system is to initialize parameters and deploy the public ones among the users of the system while storing the private parameters locally.

Public keys will be a concatenation of the dz-expression, the letter 'P' or 'S' to distinguish publisher and subscriber respectively and an epoch for key revocation. This string will be the public key and can be assembled by every user without interacting with the key server.

Private keys are generated by the key server, if the right credentials were provided during key generation.

3.2.4 Binding subscriptions

With more than one subscription or when regarding two colluding peers, the problem arises that a malicious peer could misuse his authorized subscriptions to read events, that are not destined for it. Assume again a weather service and two subscribers. The two subscriptions are $f1 = \{temperature = [10,20] \wedge humidity = [20,50]\}$ and $f2 = \{temperature = [20,30] \wedge humidity = [60,70]\}$ and the peer has got credentials for each attribute separately. $f3 = \{temperature = [10,20] \wedge humidity = [60,70]\}$ and $f4 = \{temperature = [20,30] \wedge humidity = [20,50]\}$ could also be decrypted by this peer. This needs to be prevented. Therefore, keys are bound together by the key server. When assigning keys, the key server chooses a random value q_i for each attribute and integrate it into each part of the key, so that $\Sigma_{i=0}^{d} q_i = q$. This value is canceled out, if all of the keys of the subscriptions are used when decrypting.

3.2.5 Security methods

The goal of this system is to create an overlay network, in which users can use publish / subscribe services for sensitive data. First, *authentication* is given by the interaction with the key server. Only who provides correct credentials receives keys and can either sign events or decrypt the messages, he subscribed for. Messages will be signed to proof the authenticity. Next, *confidentiality* is improved compared to other systems. Events are confidential due to encryption. Only authorized subscribers are able to decrypt. Subscriptions are not given into the system, as it is usually needed to route the events. By the use of subscription clustering and the hierarchical overlay topology some information is leaked, but we can still apply the weaker subscription confidentiality, which is already a good improvement. Third, in contrast to other approaches for confidentiality, like PKI, this work is *scalable* for large networks. The key server only maintains a single private key, no matter how many users there are in the system. Publisher and subscriber need to store a logarithmic number of keys with respect to the depth of the decomposition tree.

3.2.6 Topology and Secure Connection Protocol

We want to reach a tree topology that is ordered by containment relationships, i. e. coarse subscriptions are near the root and finer ones are further down to the leaves. In addition, a parent can only add another peer as a child, if the parent's subscription contains the subscription of the child. This means, the parent can decrypt all of the messages, that his children can decrypt, maybe more.

To achieve this and try to keep as much confidentiality as possible, a *secure connection protocol* is established. The pseudocode can be seen in algorithm 3.1.

Algorithm 3.1 Secure connection protocol at peer s_q

```
upon event: Receive (CR of s_new from s_p) do
  if decrypt_request(CR) == SUCCESS then
    if degree(s_q == available) then
      connect to s_new
    else
      forward CR to {child peers and parent} - s_p
    end if
  else[decrypt_request(CR) == FAIL]
    if s_p == parent then
      Try to swap by sending its own CR to s_new
    else
      forward CR parent
    end if
  end if
```

As a first step, a random secret word is encrypted with all public keys, that may decrypt the own subscription. If a peer wants to subscribe to "00" for instance, it is encrypted with the keys for "00" and "0". To prevent any attackers to retrieve information by looking at the number of ciphertexts, dummy messages are created. These ciphertexts will be part of a connection request CR, which is sent to a random peer s_q in the running system.

If a CR is received, the peer will try to decrypt the ciphertexts. If it succeeds, it tries to add the new peer as a child. If this is not possible, the CR needs to be forwarded in both directions, up and down the tree, as all of the addressed peers could also decrypt the CR. If the CR cannot be decrypted, the children of the peer also won't be able to decrypt, so the request needs to be forwarded to the parent, except if the request was already sent from the parent p. In this case, the peers will try to swap: the current child, say c, sends and CR to the joining peer (j). If j can decrypt CR_c, then he adds c as his child and connects to p itself (The structure will be: p -> j -> c). In case no child can swap, the joining peer needs to be added anyways as sibling to c. Note, that reconnecting one child could lead to an endless loop, especially at the beginning of the algorithm, when there are not many (or even just one possible) peer to connect to.

This procedure will be repeated for all dz-expressions and all attributes separately. A peer with multiple subscriptions will be connected multiple times in the same tree.

Worth mentioning is, that there are other approaches by Tariq[26, 27, 29] that focus different problems, in this case QoS requirements are specified in the subscrip-

tions and the topology is created to fit these. [30] adapts techniques from spectral graph theory to cluster peers in content-based publish/subscribe systems.

3.2.7 Weaker subscription confidentiality

As could be already seen in the previous sections, it is impossible to achieve strong subscription confidentiality in this system, maybe even not in broker-less publish/-subscribe systems at all. During the connection process and deduced from the topology, peers know or at least can have some guesses at the subscriptions of the other peers, especially the own parent and children. Thus, a weaker notion of subscription confidentiality is proposed in [28]:

Let s_1 and s_2 denote two subscribers in a publish/subscribe system which both possesses credentials for an attribute A_i. Weak subscription confidentiality ensures that at most the following information can be inferred about the credentials of the subscribers:

1. *The credential of s_1 is either coarser or equal to the credentials of s_2.*
2. *The credential of s_1 is either finer or equal to the credentials of s_2.*
3. *The credentials of s_1 and s_2 are not in any containment relationship.*

3.2.8 Event dissemination

As a publisher, who wants to send an event, does not know, who subscribed to the news he is going to give into the system, he needs to encrypt the message with every possible public key. This means, there will be a logarithmic amount (= length of the dz-expression) of ciphertexts for each event on each attribute tree. To reduce the load, the message is encrypted with a symmetric encryption algorithm like AES and only the symmetric key is encrypted multiple times. To check later, if the decryption was successful, a fixed length of '0' or the hash of the key should be appended. In addition sequence numbers should be assigned, so that the parts of the message from the different trees can be reassembled easily.

The initial message is sent to any random peer on each tree. A peer, that receives an event, tries to decrypt the ciphertexts. The further dissemination is similar to the secure connection algorithm shown in 3.1. If the decryption fails, the event is only forwarded to the parent, unless the parent was the sender, then it is dropped. If the decryption is successful, the peer sends the event to parent and all of his children, again as before, except the sender of the event, so that we do not have to deal with duplicates. Note, that not all of the children necessarily need to be able to decrypt the message, but due to this *one-hop flooding*, a peer does not need to have a list of subscriptions of his children, at the cost of false positives.

As soon as a peer received all parts of the message, he tries to decrypt the whole event and checks its authenticity.

3.3 Problem statement

As there most probably is no perfect system, there are some drawbacks and improvement suggestions for this one. I will explain them in the following section.

3.3.1 Key management

Currently, events and subscriptions respectively need to specify each of the d attributes, thus the number of keys to maintain is dependent on d. This is not desired, as the publish/subscribe system can support multiple independent topics. The need to declare a stock symbol when publishing an medical topic would be irritating and could make subscription matching difficult.

In addition the computation time for en-/decrypting, signing and verification are also dependent on d. When developing a technique like mentioned above, it should be asserted, that all operations only depend on the number of actually used attributes, which will be a small subset.

Therefore we need a mechanism, that allows an arbitrary number of attributes for events and subscriptions. We need to agree on a protocol, when an event matches a subscription.

3.3.2 Attacks by publishers

Like explained for subscribers in 3.2.4, publishers can misuse their credentials to publish events they are not allowed to. But in contrast to subscribers, private publisher keys are not bound together.

3.3.3 Event dissemination

One-hop flooding has the advantage, that peers do not need to know the subscriptions of their children by simply assuming, they have the same as the parent itself. In worst case in 50% of the cases, the children will not be able to decrypt the event and they will drop the message. This overhead is called *false positives* and should be minimized, as it causes a lot of unnecessary traffic and computation cost at the peers.

4 Analysis of subscription confidentiality

Last chapter described the approach developed by Tariq et al.[28]. Now we will analyze the threat of attacks to break the confidentiality of this system, state a formula for measuring confidentiality and discuss an alternative to one-hop flooding. Evaluations will be given afterwards.

4.1 Problems with the current approach

There are two main issues we will focus.

1. First, as described in 3.2.7, a weaker subscription confidentiality is assumed, as a strong confidentiality probably cannot be provided. Clustering subscriptions already reveals some information about containment relationships, thus, restrictions are eased. We will simulate an attack scenario to examine the decrease of confidentiality.

2. The second one is a performance issue. One-hop flooding renders the need to know one children's subscriptions unnecessary, but generates a lot of false positives on the other hand, which are undesirable. We will introduce a technique named *k-subscriptions*, that allows us to drop one-hop flooding with only a slight change in the system, so that not much confidentiality is lost, but we reduce the false positives to a minimum.

4.1.1 Traffic analysis

Raymond[17] proposes some methods for traffic analysis. With such techniques an attacker can gather some information. Even if we assume point-to-point encryption, where the bit string of a forwarded event changes on each hop, messages can be traced by means of timing attacks, as links usually have delays in some estimated bounds. Just as soon as there are communication links, that the adversary cannot access, it must assume different clusters. If some malicious node can decrypt an event and communicates the topic to the attacker, it can deduce information about other peers in the same cluster. If a peer forwards a message to its children, the adversary knows that it must be able to decrypt the message and can store this in a knowledge base. Subsequent notifications can improve this. The other way round it can be determined if the message cannot be read, when it is

dropped. Even the weaker subscription confidentiality 3.2.7 can be limited a bit. As soon as the last explained case occurs, the attacker (or even the child itself) can state, that the child has a coarser (and not equal!) subscription then the parent. Similarly during the connection protocol, if a connection request is forwarded to a parent and no children, this either means that the peer cannot take more children or is not able to decrypt the request. But if the parent can decrypt the request, it can discover that the child got a finer (and not equal!) subscription.

As a last example, fake events can be used to detect subscriptions, as they are only dropped after they are successfully decrypted. When sending a fake message to a branch, it will be forwarded upwards until some peer decrypts and drops it. This will reveal some evidence about the peers. Repeating with different events can even break some subscriptions completely.

4.2 Attack scenario

In this scenario the attacker is omnipresent, meaning he knows or can observe the protocol and network parameters. Thus it knows the communication delay of the links, can eavesdrop on them and with these information trace messages sent through the network with all means of traffic analysis, as explained for instance by Raymond[17]. As soon as any message is detected on a communication link, the receiver will be targeted, to notice on which link the event is forwarded. As long as all links are visible for the attacker, the message can be tracked during its whole lifetime. Note, that this is also possible, if point-to-point encryption is used and the bit sequence of the message changes on each hop. Knowing the underlying topology and communication delays is enough to keep track. In the following, we will assume, that not all links will be visible, as it is a reasonable assumption, because it will be very hard for an attacker in the real world to control all of the links. In this case, the attacker will loose track of the message, as soon as it passes a communication link, that is not observable. The attacker can only guess how many peers the event has passed, since it lost track as soon as it reappears but needs to assume, that it might be a new message. In this way, the topology will be divided into clusters in attacker's view and each cluster needs to be handled separately.

Additionally there will be some malicious nodes, that support the adversary. This can be peers, the attacker set up by itself or regular users, that were compromised. By all means, they will all collude to reveal as many information as possible.

4.3 k-subscriptions

As already mentioned, false positives are to be avoided. To keep confidentiality high, we propose *k-subscriptions* which idea is similar to k-anonymity[20].

As a first step, we will simply include the information about the subscription of their children in each peer. Then, an events can be forwarded only to the children, that are interested in the message. Due to the clustering, there will be no more false positives as soon as the notification reaches a peer, that is able to decrypt.Then we make sure, each peer has got at least "k" subscriptions by expanding them. Recall the technique to decompose the event space using dz-expressions. Assume, $k = 3$ and an initial dz-expression of {1} as subscription. In the first step, this can be expanded to the two expressions {10, 11}. Then another expansion may result in {10, 110, 111}. The covered event space is still the same, but there are three dz-expressions instead of only one. Now, when the peer connects to the network, it should be assured, that each dz-expression is connected to a different parent. To assert, that expansion is possible, a maximum granularity needs to be declared.

4.4 One-hop flooding and k-subscriptions

The original approach uses one-hop flooding, what is intuitively a good idea, because parents do not need to know the subscriptions of their children and always forward events that could be decrypted to them. Still it is obvious, that there will be many false positives. Theoretical worst case value should be 50% of false positives. Thus we will analyze the idea to drop one-hop flooding to improve performance.

4.5 Measurement of confidentiality

To compare different strategies, we will use two different formulas as both have their right to exist. Both are calculated in the same way:

$$C = 100 * \frac{compromised\_peers - malicious\_nodes}{total\_peers - malicious\_nodes}[in\%]$$

They differ only in the definition of compromised peers. We denote C_{all} as the confidentiality where a peer is considered compromised only if *all* of its subscriptions on all trees are revealed by the attacker. So this is an upper bound for the values and is expected to be quiet high. In the second case, we count a peer to the compromised, as soon as *any* information is leaked. Intuitively we can already anticipate, that this definition will yield much smaller numbers. Without one-hop flooding, the subscriptions of children of malicious nodes will always be seen as compromised. In combination with the C_{any} measurement, it will be interesting to see, how many subscriptions are exposed per peer.

In general we can state, that the following inequality will hold, because the first definition is a special case of the second, where on *each* compromised peer all subscriptions are discovered, and not only a part of them.

$$C_{all} >= C_{any}$$

4.6 Evaluation

In this section, we measure the confidentiality of Tariq's work[28] and compare it with k-subscriptions.

System specific attack details

On the one hand malicious nodes reveal their own subscription. This is why they are excluded, when calculating confidentialities in section 4.5. As soon as we do not use one-hop flooding anymore and the subscriptions of the children will be known to the parent (explained in 4.4), the children's subscriptions are also revealed. On the other hand, they tell the adversary, whether they can decrypt events, they received or not. If so, they will also inform the attacker about the contents. Of course, this will only be valid in the cluster, the malicious node belongs to. Combined with the information, if a peer forwarded a message to its children, a list can be maintained for each peer that records the events, which that peer can decrypt. Similarly, a list of all events that cannot be decrypted will be kept. Aggregating multiple entries will eventually lead to a guess of the real subscription.

As an example, we look at the graph in figure 4.1. The green arrows depict, where the event is sent to. Here, node "100" is the publisher and sends the first message to a random node in the network ("0"). This cannot decrypt the event "111" and thus forwards the message only to its parent. When one-hop flooding is enabled, the attacker can now conclude correctly, that the event was not decrypted. Later the event reaches node "1". As it forwards the message to its children, it needs to be able to read the contents, but only one of its child can, the other not (red arrow). The latter produces a false positive. If the subscriptions would be known to the parents, this red message would not have been sent.

The algorithms, the attacker executes if an event is recognized at a specific peer is given in figures 4.1 and 4.2 in pseudocode. They depend on the observation, if the message is forwarded and the information, any malicious node present in the corresponding cluster can provide. As can be seen, they do not differ much. In case of k-subscriptions, the subscriptions of children are also revealed. As a trade-off no information can be inferred, if the event was not forwarded to any child. Keep in mind, that this even means an improvement in confidentiality in some cases. When using one-hop flooding, this always means, that the event could not be decrypted. In k-subscriptions it is also possible, that no child can decrypt the message.

Experimental setup

We will simulate a publish / subscribe network with up to 2048 peers in PeerSim[12]. If not otherwise mentioned, d = 4 attributes are used with integer values in the range [1, 16]. The number of malicious nodes vary between 5% and 25% and

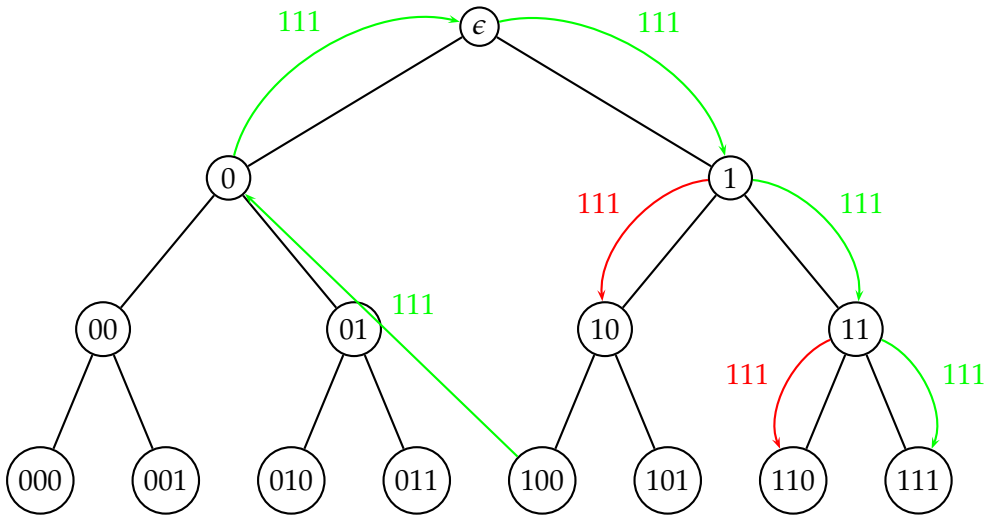

Figure 4.1: Exemplary attack move

Algorithm 4.1 Function EvalAttack for one-hop flooding

if Peer is malicious **then**
    Reveal subscription of this peer and children (add to corresponding positiveLists)
end if
if Event is known to attacker AND Event was forwarded to any child **then**
    Add Event to positiveList
end if
if Event is known to attacker AND Event was NOT forwarded to any child **then**
    if Children are present **then**
        Add Event to negativeList
    else[no children are present]
        Nothing can be inferred
    end if
else if Event not known to attacker AND Event was forwarded to any child **then**
    Add all subscriptions of malicious nodes in this partition to negativeList
end if

Algorithm 4.2 Function EvalAttack for k-subscriptions

```
if Peer is malicious then
    Reveal subscription of this peer(add to corresponding positiveLists)
end if
if Event is known to attacker AND Event was forwarded to any child then
    Add Event to positiveList
else if Event not known to attacker AND Event was forwarded to any child then
    Add all subscriptions of malicious nodes in this partition to negativeList
end if
```

there are at least 20% and maximum all links observable. For simulations with k-subscriptions, the variable was chosen as k = 3.

For both, subscription generation and event generation zipfian distribution is used, as it is more realistic than uniform distribution. At first, all peers are added and connected and in a second phase, events are generated. Attack evaluation is called at each peer after forwarding the notification.

All results are the average of five runs.

Varying number of attributes

At first will have a look at simulation results for different number of dimensions.

We choose a hostile environment, so that C_all will not be too high. There will be 25% malicious nodes and 80% links visible while looking at d = 2 up to d = 16 dimensions.

In 4.2 we can see, that the behavior of both approaches are very similar. Here, C_{all} is at about 87% confidentiality, which rises up to almost 100% for eight and more trees.

d = 4 seemed to be a good value for further simulations, as it is not too close to 100% but it is still a reasonable choice.

Fixed malicious nodes

Figure 4.3 displays the difference for an fixed value of 5% of malicious nodes. While k-subscriptions even performs better for C_{all} and when more then about 50% of links are visible, one-hop flooding reaches about 80% confidentiality in contrast to only 40% for k-subscriptions with 20% links visible. Why k-subscriptions can outperform one-hop flooding in some cases was already explained in chapter 4.6.

Still, we need to look at the number of discovered subscriptions to get a real impression. In figure 4.4 the distribution is shown for 5% malicious nodes and k = 3, meaning, each peer has at least three subscriptions per dimension, thus more

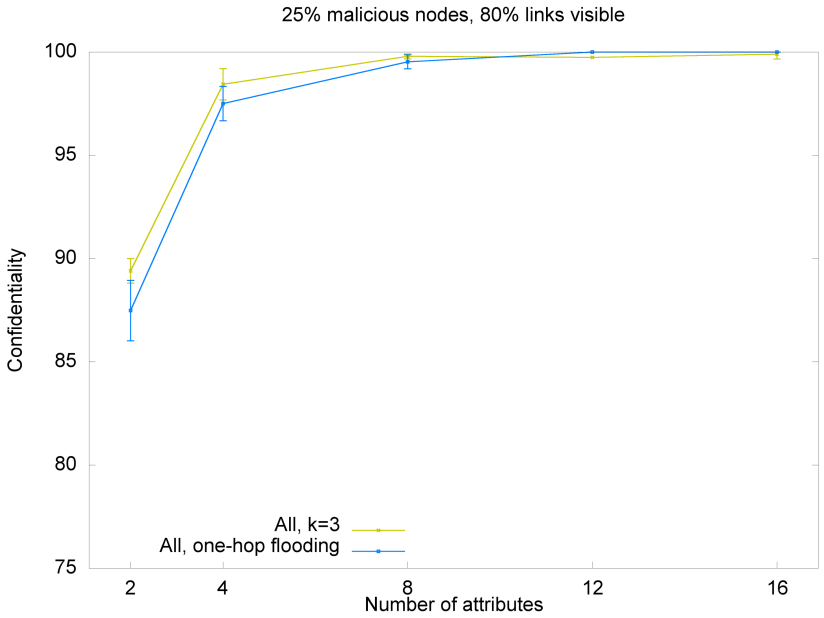

Figure 4.2: 25% of malicious nodes, 80% of links visible

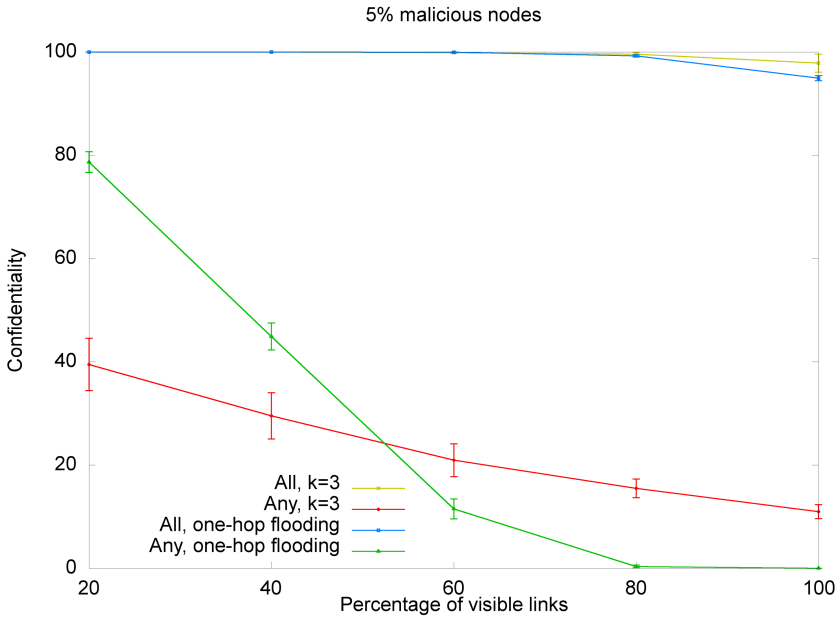

Figure 4.3: Comparison of one-hop flooding and k-subscriptions with 5% malicious nodes

subscriptions altogether. This example yields an average number of subscriptions of 13 per peer.

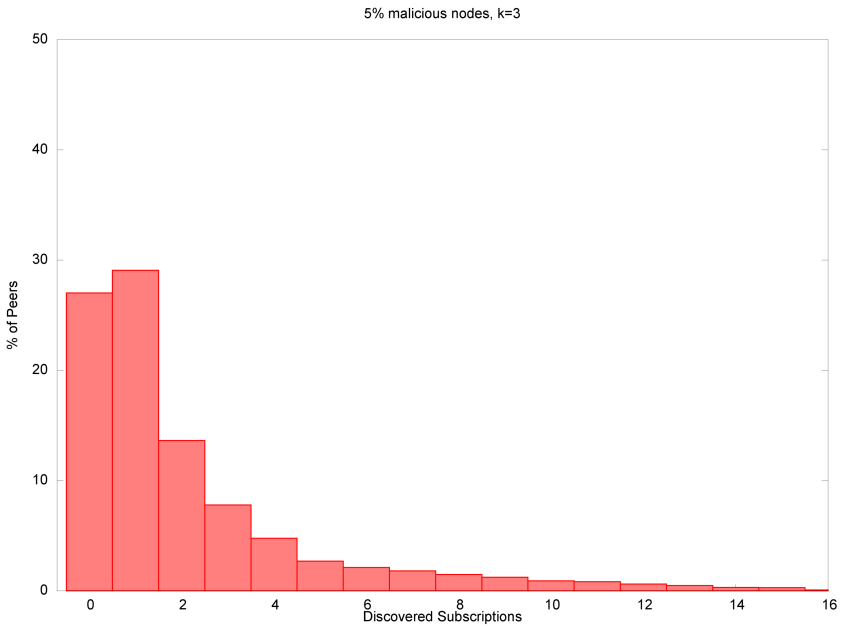

Figure 4.4: Discovered subscriptions per peer with k=3 subscriptions

In contrast to this, figure 4.5 depicts the same scenario with one-hop flooding. It looks better at first glance, but there are only an average of about 8.5 subscriptions per peer. This means in the first case about 50% more subscriptions need to be compromised to extract the same amount of information. Looking at the figures regarding this, we recognize, that in both scenarios, the most peers only got one compromised subscription. If we look at k-subscriptions at least three subscriptions need to be compromised to have the full information about one attribute. The probability, that the first three discovered subscriptions all belong to the same attribute is very low. Thus, the result is very promising.

Figure 4.6 illustrates the run of the curve for a high value of 25% of malicious nodes. As expected, C_{all} is still always near 100%, verifying, that it is hard to break all of the subscriptions. Only if the number of links is also very high or even all are observable by the attacker, confidentiality drops to about 92%, which still is a good value for that scenario. Only C_{any} is really low. But we cannot expect to keep a strong confidentiality in such an unfriendly environment. But for only a view visible links with one-hop flooding there is a good portion of confidentiality preserved.

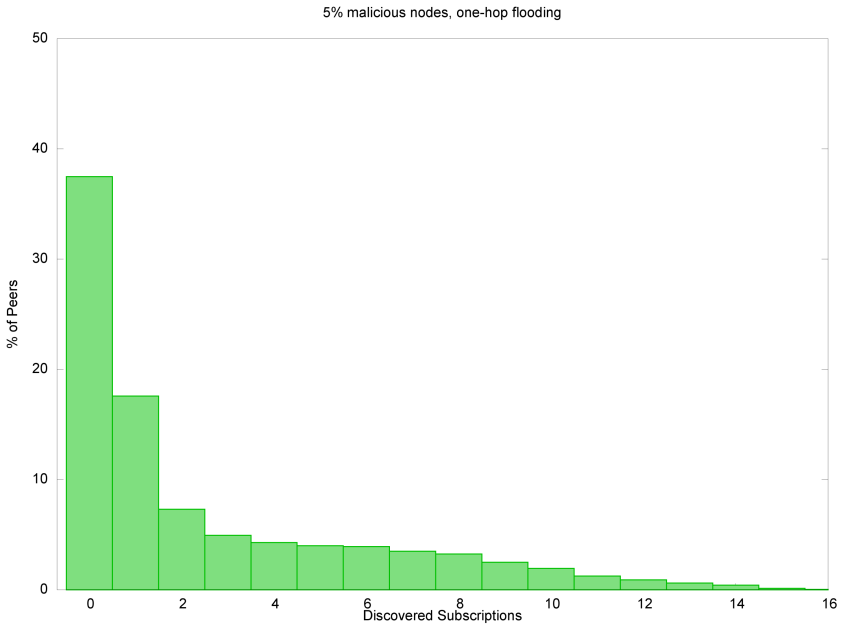

Figure 4.5: Discovered subscriptions per peer with one-hop flooding

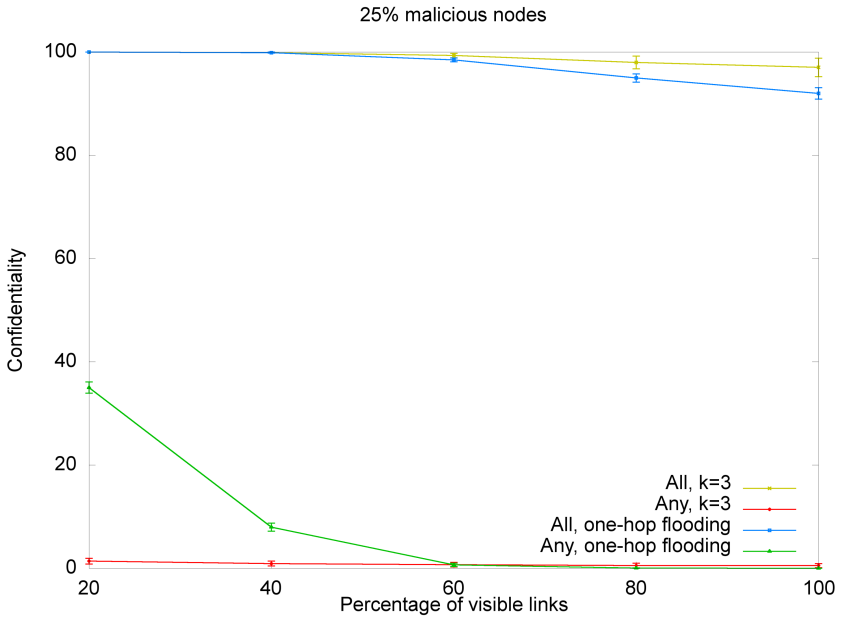

Figure 4.6: 25% of malicious nodes

Fixed link visibility

With 20% of visible links (4.7) one-hop flooding beats k-subscriptions easily. Still the distribution of compromised peers in 4.8 looks fine so that k-subscriptions cannot be ruled out.

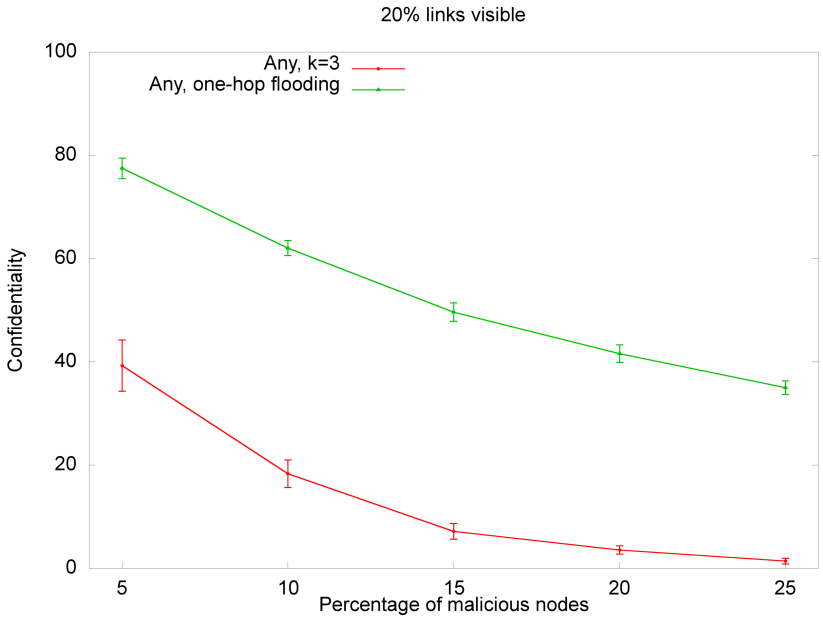

Figure 4.7: 20% of links visible

When keeping the percentage of links visible to the attacker fixed at 40% the results shown in 4.9 are once again in favor of one-hop flooding. The curve for one-hop flooding is constantly about 8 to 15 percentage points higher.

Summary

The confidentiality, that can be achieved by this system is very good. Of course, it is hard to defend privacy in hostile environments with a lot of malicious nodes and observable links, but in most scenarios only few subscriptions per peer are compromised. Yet, the weaker notion of subscription confidentiality cannot be met.

We have seen, that k-subscriptions is a reasonable approach and that not too much confidentiality is lost and in some cases even gained a bit. Furthermore the advantage is huge: Whereas one-hop flooding yields about 22-30%, in k-subscriptions there are only an average of 2% false positives with a peak value of 5.5%. These still occur, because the events are initially sent to a random peer. If this one is not able to decrypt the event by chance, it will be forwarded further up the tree, until any

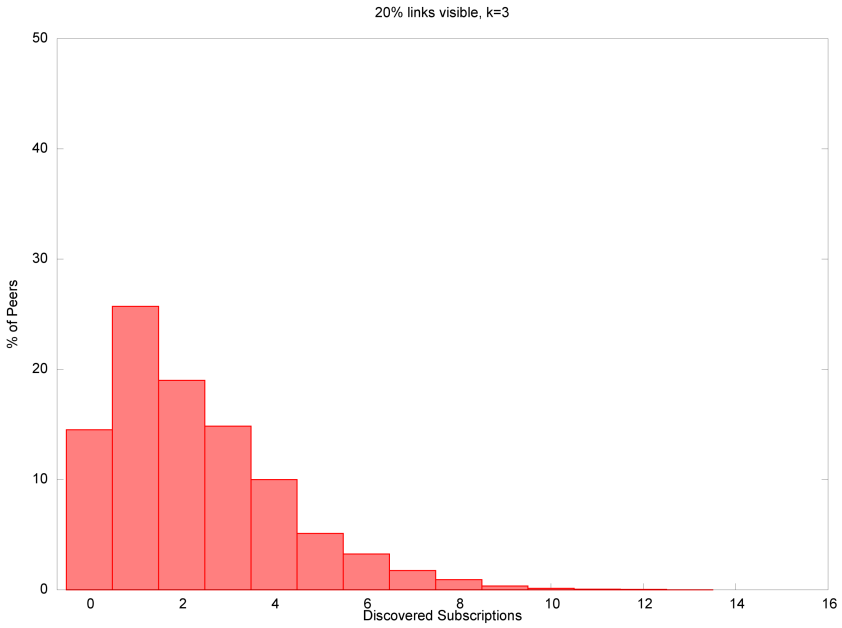

Figure 4.8: Discovered subscriptions per peer with k-subscriptions

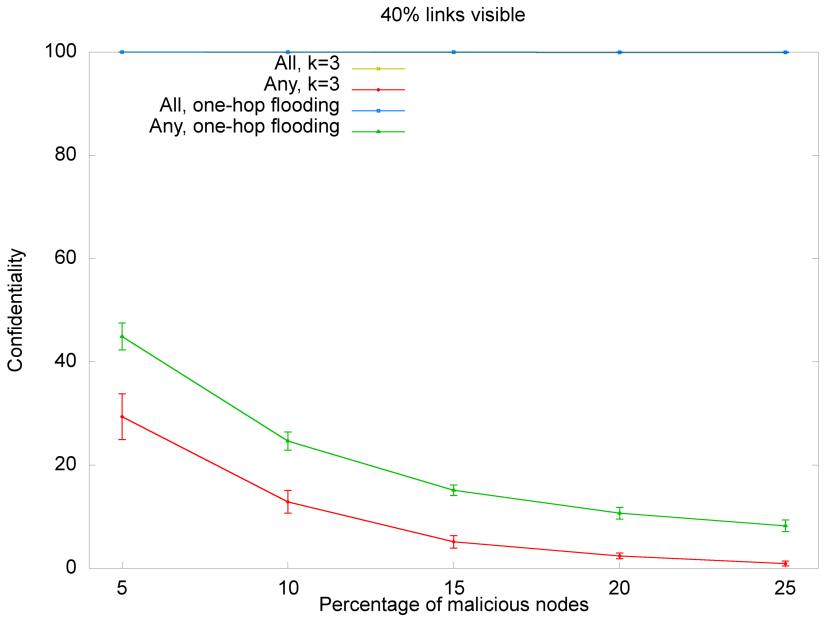

Figure 4.9: Comparison of one-hop flooding and k-subscriptions with 40%links visible

peer can decrypt the event. All of the nodes on this way will count the event as a false positive. So we cannot expect to reach much better values by other methods. Through this reduction performance should increase greatly.

5 Cryptographic alternative

In this chapter, I will present an alternative cryptographic scheme, that is a combination of the signcryption scheme proposed in [33] and attribute-based encryption like explained in [7] and [18].

There are two goals:

- Reduce attacks from publishers, as described in section 3.3.2. Keys need to be bound together to prevent malicious publishers to mix their private keys to publish events, they are not allowed to.

- Improve key management. As stated in section 3.3.1, the number of keys to maintain for publishers as well as subscribers is dependent on the size of the event space. The algorithm provided in this chapter will reduce the keys to be dependent on the number of used attributes only.

In the current approach, all attributes must be specified on each subscription and each advertisement or event. If we take a newsgroup as example the topic variety is possibly huge. Yet, if we want to publish a news about the whether it is meaningless, if we have to specify a stock quote. As explained in 2.1, attribute-based encryption enables fine-grained access control. This technique will be used to decrypt messages, where the advertisements and subscriptions are not an exact match.

For an event with two attributes, let us say $E = \{a = 4 \wedge b = 2\}$, not only exact subscriptions like $f_1 = \{a > 3 \wedge b < 7\}$ but also any subscription like $f_2 = \{a > 3\}$ and $S_3 = \{b < 7\}$ should match. With a system like this it would not be necessary to declare all attribute of the universe.

We will see, that efficiency is improved coherently. Timings will not be dependent on the number of attributes of the universe but only of the actually used attributes.

Like in most identity-based encryption schemes, we will divide this approach into four steps: Setup, key generation, encryption and decryption. In this case, encryption and decryption will be replaced by signcryption and unsigncryption to include authentication.

5.1 Setup

During the setup phase, the key server calculates and chooses the private and public system parameters needed for further progress. Like in the other papers, bilinear maps are used, so they need to be set up at first.

Now let $\mathbb{G}_1$, $\mathbb{G}_2$ be cyclic groups of prime order p with $|\mathbb{G}_1| = |\mathbb{G}_2| = p$, e be an admissible bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$. g shall be a generator of $\mathbb{G}_1$. Next we need two collision resistant cryptographic hash functions $H_1 : \{0,1\}^* \rightarrow \{0,1\}^{n_u}$ and $H_2 : \{0,1\}^* \rightarrow \{0,1\}^{n_m}$ with hash length n_u and n_m respectively. We need two vectors $\vec{u}$ and $\vec{m}$ of the same length n_u and n_m. Its elements u_i and m_i are random elements from $\mathbb{G}_1$. Coupled to these vectors we need another random element for each, say u' and m', again from $\mathbb{G}_1$.

Now a number $\alpha \in \mathbb{Z}_p$ is chosen randomly. Then $g_1 = g^\alpha$, g_2^α and $Y = e(g_2, g)^\alpha$ are calculated.
The Master Private Key, which is only known to the key server and should be kept secret is g_2^α. The Master Public Key consists of $(e, g, g1, g2, u', m', \vec{u}, \vec{m}, Y)$. This key is distributed in the system and available for every peer. It is used for encrypting events and verifying its signature.

5.2 Key Generation

Key generation is performed, as soon as a peer contacts the key server. The peer needs to provide its credentials and the key server decides if they are correct. As already stated in , the credentials consist of capabilities and authentication and we only want to care about the former one. Let $Cred_{i,j}$ be the credential for attribute A_i with label j with.

The public keys are generated by a simple concatenation

$$Pu_{i,j}^p := (Cred_{i,j}||Ai||P||Epoch)$$

for publishers and

$$Pu_{i,j}^s := (Cred_{i,j}||Ai||S||Epoch)$$

for subscribers. Only one flag needs to be switched and this can be done by each peer itself. No interaction with the key server is necessary.

Private keys are more interesting to look at. For publishers, we need to choose a random value $r_{i,j} \in \mathbb{Z}_p$, calculate $r = \Sigma_i r_{i,j}$, the hash bit string $v_p = H_1(Pu_{i,j}^p)$ of length n_u, where $v_p[k]$ is the k-th bit of this string. We denote $\Gamma_{i,j} \subseteq \{1, 2, \ldots, n_u\}$ as the set of all indexes k for which the k-th bit of the hash $v_p[k] = 1$.

$$Pr_{i,j}^p := (Pr_{i,j}^p[1], Pr_{i,j}^p[2])$$

with

$$Pr^p_{i,j}[1] = g_2^{\alpha}(u'\Pi_{k\in\Gamma_{i,j}}u_k)^{r_{i,j}}g^{r_{i,j}}$$
$$Pr^p_{i,j}[2] = g^{r_{i,j}}$$

Similarly, the key server chooses for the subscriber at random the values r_s and $r = \Sigma r_{i,j}$. $\Gamma_{i,j}$ is calculated as in the publisher's case. Then the private keys are:

$$Pr^s_{i,j} := (Pr^s_{i,j}[1], Pr^s_{i,j}[2])$$

$$Pr^s_{i,j}[1] = g_2^{r_{i,j}}(u'\Pi_{k\in\Gamma_{i,j}}u_k)^{r_s}$$

and two attribute independent parts

$$Pr^s_{i,j}[2] = g^{r_s}$$
$$Pr^s[3] = g_2^{\alpha+r}$$

They will be needed to make sure, all of the subscriptions are used and thus, they are bound together.

5.3 Signcrypt

Again, choose a random $q \in \mathbb{Z}_p$.

The ciphertext σ will consist of multiple parts.

$$\sigma_1 = C_1 = MY^q$$

Note, that as in the original approach it might be more suitable to encrypt only a symmetric key, which is used to encrypt the actual message. In the following, let $\Gamma_{i,j}$ be defined as in the previous section for key generation and Γ_m analogous, but with $v_m = H_2(M)$ the message hash.

$$\sigma_2 = C_2 = g^q$$
$$\sigma_{3i} = C_i = (u'\Pi_{k\in\Gamma_{i,j}}u_k)^q)$$
$$\sigma_{4i} = C^{sign}_{i,j}[1] := Pr^p_{i,j}[1](m'\Pi_{k\in\Gamma_m}m_k)^q)$$

finally

$$\sigma_5 = C^{sign}_{i,j}[2] := Pr^p_{i,j}[2]$$

$\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5)$ will be send into the network. It is still necessary to send a logarithmic amount of ciphertexts, one for each credential, that shall able to decrypt the message.

5.4 Unsigncrypt

To decrypt the message, a peer needs private keys which credentials provide them access. The following steps have to be performed:

$$\begin{aligned}\frac{e(Pr^s_{i,j}[1], \sigma_2)}{e(Pr^s_{i,j}[2], \sigma_{3i})} &= \frac{e(g_2^{r_{i,j}}(u'\Pi_{k\in\Gamma_{i,j}}u_k)^{r_s}, g^q)}{e(g^{r_s}, u'\Pi_{k\in\Gamma_{i,j}}u_k)^q)} \\ &= \frac{e(g_2^{r_{i,j}}, g^q)e((u'\Pi_{k\in\Gamma_{i,j}}u_k)^{r_s}, g^q)}{e(g^{r_s}, u'\Pi_{k\in\Gamma_{i,j}}u_k)^q)} \\ &= e(g_2, g)^{r_{i,j}q}\end{aligned}$$

Multiplying $e(g_2, g)^{r_{i,j}q}$ for each subscription results in

$$\Pi_{i=1}^{Subs} e(g_2, g)^{r_{i,j}q} = e(g_2, g)^{rq}$$

Now compute:

$$\begin{aligned}\frac{e(\sigma_2, Pr^s[3])}{e(g_2, g)^{rq}} &= \frac{e(g^q, g_2^{\alpha+r})}{e(g_2, g)^{qr}} \\ &= \frac{e(g, g_2)^{q\alpha+qr}}{e(g_2, g)^{qr}} \\ &= e(g2, g)^{q\alpha}\end{aligned}$$

Finally the message can be extracted by

$$\begin{aligned}\frac{\sigma_1}{e(g2, g)^{q\alpha}} &= \frac{MY^q}{e(g2, g)^{q\alpha}} \qquad\qquad = \frac{Me(g2, g)^{q\alpha}}{e(g2, g)^{q\alpha}} \\ &= M\end{aligned}$$

The message shall only be accepted, if an authorized publisher disseminated the event. Thus, after decrypting a verification procedure follows.

Verify that

$$\frac{e(\sigma_{4i}, g)}{e(g1, g2)e(u'\Pi_{k\in\Gamma_{i,j}}u_k, \sigma_5)e(m'\Pi_{k\in\Gamma_m}m_k, \sigma_2)} = e(g, g)^s$$

with

$$e(g, g)^s = \Pi e(g^{r_{i,j}}, g)$$

As proof for the correctness, follow the next equations for each advertisement.

$$\frac{e(\sigma_4, g)}{e(g1, g2)e(u'\Pi_{k\in\Gamma_{i,j}} u_k, \sigma_5), e(m'\Pi_{k\in\Gamma_m} m_k, \sigma_2)}$$

$$= \frac{e(g_2^{\alpha}(u'\Pi_{k\in\Gamma_{i,j}} u_k)^{r_{i,j}} g^{r_{i,j}}(m'\Pi_{k\in\Gamma_m} m_k)^{q}, g)}{e(g1, g2)e(u'\Pi_{k\in\Gamma_{i,j}} u_k, g^{r_{i,j}}), e(m'\Pi_{k\in\Gamma_m} m_k, g^{q})}$$

$$= \frac{e(g_2^{\alpha}, g)e((u'\Pi_{k\in\Gamma_{i,j}} u_k)^{r_{i,j}}, g)e(g^{r_{i,j}}, g)e((m'\Pi_{k\in\Gamma_m} m_k)^{q}, g)}{e(g1, g2)e(u'\Pi_{k\in\Gamma_{i,j}} u_k, g^{r_{i,j}}), e(m'\Pi_{k\in\Gamma_m} m_k, g^{q})}$$

$$= \frac{e(g_2, g^{\alpha})e((u'\Pi_{k\in\Gamma_{i,j}} u_k), g^{r_{i,j}})e(g^{r_{i,j}}, g)e((m'\Pi_{k\in\Gamma_m} m_k), g^{q})}{e(g1, g2)e(u'\Pi_{k\in\Gamma_{i,j}} u_k, g^{r_{i,j}}), e(m'\Pi_{k\in\Gamma_m} m_k, g^{q})}$$

$$= e(g^{r_{i,j}}, g)$$

5.5 Evaluation

Simulations show the timings of this scheme. Measurements are done on an Intel Core i5-2500K@3.3GHz processor with 16 GB of memory using minGW on Windows 7. The algorithm was implemented in C++ with the Pairing-Based Cryptography library[10]. A bilinear map based on a supersingular elliptic curve $Y^2 = X^3 + X$ and a base field size of 512 bit is used.

The timings seen in table 5.1 are a result of the average of 10'000 runs.

$Time_E$	3.22350 + d' * 3.72690
$Time_S$	d' * 3.91136
$Time_D$	3.24051 + dz * 6.43670
$Time_V$	3.24357 + d' * 7.34714

Table 5.1: Computation times (in msec) of the new approach

d' denotes the number of used attributes and dz the number of subscriptions for the peer. I want to stress again, that the timings are no longer dependent on the total number of attributes d, the system uses and it maybe $d' \ll d$.

5.6 Security

In this approach, a weaker notion of subscription confidentiality is provided as in the original scheme, by clustering peers according to their subscriptions in a hierarchical tree for each attribute. In case of k-subscriptions, each peer ensures, that it has at least k subscriptions and for each of them a different parent (whenever possible). Strong subscription confidentiality cannot be reached in this manner,

because by design it is given, that parents know, that their children must have a finer or equal subscription or even, unless one-hop flooding is used, know the subscriptions of their children.

The scheme described in this chapter is used to provide event-confidentiality. An identity-based signcryption scheme is used and extended with the advantages of attribute-based encryption to be able to use only a subdimension for events and subscriptions.

The identity based signcryption scheme [33] was already proved indistinguishable against adaptive chosen ciphertext attacks, even without random oracles.Semantic security is based on the hardness of the Decisional Bilinear Diffie-Hellman problem and the Computational Diffie-Hellman assumption was used to prove unforgeability.

Key-policy attribute-based encryption was reduced to the hardness fo the Decisional Bilinear Diffie-Hellman assumption.

This is a solid basis for a new cryptosystem. In the encryption process not much was changed. Adaptions were made to allow multiple dimensions. For the signature, the factor $g^{r_{i,j}}$ was added to the publisher's private key $Pr^{p}_{i,j}[1]$. Still, the presence of the master public key g_2^{α} prevents forgery.

6 Conclusion

6.1 Summary

Aim of this thesis was to review [28], where mechanisms were introduced to provide authentication, publisher and subscriber confidentiality in broker-less content-based publish / subscribe systems. Analyzing this approach especially with respect to confidentiality and finding other shortcomings and solutions for these followed.

To be able to understand the scheme, we first looked into some background. Bilinear maps in the context of identity-based encryption were introduced. IBE uses unique names for users as public keys for asymmetric encryption. Four steps are necessary for communication: 1) Setup, where the key server determines and calculates system parameters, keeps the master secret key private while publishing the master public key, 2) key generation as soon as a peer contacts the peer server and authenticates itself, a private key for this user is generated. As the public key is a string, that can be easily obtained by any peer, no interaction with the key server is necessary to write and encrypt a message as step 3). Lastly, step 4) is the decryption of the message with the private key.

The approach itself proposes to cluster peers according to their subscriptions. The event space is decomposed by creating dz-expressions. The subscriptions are mapped to binary strings by continuous bisection. For each bisection one bit is appended to the dz-expression: “0” for the lower half, “1” for the upper one. Using this technique, containment relationships can be determined easily by prefix matching. For each attribute of the publish / subscribe system, the peers are structured in hierarchical trees by the secure connection protocol. Parents always have coarser or equal subscriptions than their children. As this exposes some information about the subscriptions, strong subscription confidentiality cannot be provided. Therefore a weaker notion of subscription confidentiality is introduced, which states, that only containment relationships can be determined.

Having this topology, event dissemination is easy. The message is encrypted for each dz-expression, that shall be able to decrypt, which yields a logarithmic number of ciphertexts. A peer receiving these ciphertexts, tries to decrypt them with the own private keys. If it is successful, one-hop flooding is used to deliver the ciphertexts to the children, as their subscriptions shall be unknown.

This approach was implemented to run some simulations using PeerSim[12]. An attack scenario explained who an attacker can break the confidentiality of the sys-

tem. The scenario assumed to have access to malicious nodes in the network, which reveal their subscriptions and decrypted events. Now notifications can be traced and by observing the network traffic, the attacker can tell, whether an event was forwarded to the children, meaning, the event was decrypted, or not. Aggregating the events that passed over a certain time may reveal the real subscription. The simulation showed, that still only parts of the subscriptions can be determined. If a peer is considered compromised only, if all of its subscriptions are known to the attacker, confidentiality is even with a high number of 25% malicious nodes with all links observable very good and only drops to about 93%. If one broken subscription is enough to call a peer compromised, the figures do not look as good at first glance. But further analysis shows, that only a small part of the subscriptions are broken on each peer.

This supports the idea of "k-subscriptions", which we initially came up with to reduce false positives. In this mechanism, we ensure that each peer's subscriptions are divided into finer ones until there are at least k subscriptions on each tree. As then not significantly more subscription parts are discovered, less information is given away. This advance is used to drop one-hop flooding, as it reduces false positives by a great number. We let peers always know the exact subscription for the connection of their children to be able to forward messages only to peers, that subscribed for the event. The benefit is a reduction of false positives from about 25% to only 2%. The remainder of false positives are inherent as a new event is always sent to a random peer to preserve publisher confidentiality. As the figures show, not too much confidentiality is lost and this mechanism can be well established.

Other drawbacks of the original approach are at first, that only subscriber keys are bound together but not publisher keys and second that events and subscriptions always need to specify all attribute, that exist in the network. As this can be a lot more, than needed and still cryptographic operations need to be executed for all of them, this means a great overhead. So a new method was proposed in chapter 5. Signcryption and attribute-based encryption were combined to cope with the requirements. Again, this approach was implemented, this time using the Pairing-Based Cryptographic Library PBC [10]. Results show reasonable timings for the cryptographic operations. Moreover they are now only dependent on the number of used attributes d' instead of the whole attribute space d.

6.2 Future work

Four major issues were handled in this thesis: Analysis of confidentiality, reduction of false positives, binding of publisher keys and the need to specify all attributes.

Still, as this field of work is quite young, it offers a lot of tasks more.

At first, it would be very interesting to look for techniques, that can provide stronger subscription confidentiality. We have discussed earlier, that we will al-

ways need a weaker subscription confidentiality, if we cluster peers. There might be a different strategy to cluster peers without leaking so much information. Perhaps a cryptofunction can be used to check, if two subscriptions are similar, without letting any of the parties know anything. Subscribers could be sorted without containment relationships, then probably not in trees. Still, this could arise the need for a new routing mechanism.

Another shortcoming so far is the need to decrypt messages before verifying its signature. Suppose an unauthorized peer, that publishes a fake message. The message is sent to a random peer initially. If it cannot decrypt the message, it will forward it to its parent. This happens, as long, as no peer can decrypt the message, in the worst case until the root. Only then it will be dropped, as the signature won't verify. Moreover, an attacker could follow this message to observe at which peer it will be dropped. Then it can state, that non of the peers on the way were able to decrypt. If an event could be verified before it is even decrypted, then fake notifications could always be dropped on the first hop.

Bibliography

[1] Group 15. Identity-based encryption. 2007.

[2] Dan Boneh and Matt Franklin. Identity-based encryption from the Weil pairing. *SIAM J. of Computing*, 32(3):586–615, 2003. extended abstract in Crypto'01.

[3] Antonio Carzaniga, David S. Rosenblum, and Alexander L. Wolf. Design and evaluation of a wide-area event notification service. *ACM Trans. Comput. Syst.*, 19(3):332–383, August 2001.

[4] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In *Proceedings of the 8th IMA International Conference on Cryptography and Coding*, pages 360–363, London, UK, UK, 2001. Springer-Verlag.

[5] Michael J. Freedman and Robert Morris. Tarzan: a peer-to-peer anonymizing network layer. In *Proceedings of the 9th ACM conference on Computer and communications security*, CCS '02, pages 193–206, New York, NY, USA, 2002. ACM.

[6] David Goldschlag, Michael Reed, and Paul Syverson. Onion routing. *Commun. ACM*, 42(2):39–41, February 1999.

[7] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM conference on Computer and communications security*, CCS '06, pages 89–98, New York, NY, USA, 2006. ACM.

[8] Mihaela Ion, Giovanni Russello, and Bruno Crispo. Supporting publication and subscription confidentiality in pub/sub networks. *Security and Privacy in Communication Networks*, 2010.

[9] Sachin Katti, Jeffrey Cohen, and Dina Katabi. Information slicing: Anonymity using unreliable overlays. *CSAIL Technical Reports (July 1, 2003 - present)*, 2007.

[10] Ben Lynn. The pairing-based cryptography library. `http://crypto.stanford.edu/pbc/`, 2007. [Online].

[11] Alfred Menezes. An introduction to pairing-based cryptography. *Lecture Notes*, 2005.

[12] Alberto Montresor and Márk Jelasity. PeerSim: A scalable P2P simulator. In *Proc. of the 9th Int. Conference on Peer-to-Peer (P2P'09)*, pages 99–100, Seattle, WA, September 2009.

[13] Gero Mühl. Large-scale content-based publish-subscribe systems. Technical report, TU Darmstadt, November 2002.

[14] Lukasz Opyrchal and Atul Prakash. Secure distribution of events in content-based publish subscribe systems. In *Proceedings of the 10th conference on USENIX Security Symposium - Volume 10*, SSYM'01, pages 21–21, Berkeley, CA, USA, 2001. USENIX Association.

[15] Lauri I. W. Pesonen, David M. Eyers, and Jean Bacon. Encryption-enforced access control in dynamic multi-domain publish/subscribe networks. In *Proceedings of the 2007 inaugural international conference on Distributed event-based systems*, DEBS '07, pages 104–115, New York, NY, USA, 2007. ACM.

[16] Peter R. Pietzuch. Hermes: A scalable event-based middleware. Technical Report UCAM-CL-TR-590, University of Cambridge, Computer Laboratory, June 2004.

[17] Jean-Franois Raymond. Traffic analysis: Protocols, attacks, design issues and open problems. In *Proceedings of international workshop on design issues in anonymity and unobservability*, pages 10–29. Springer-Verlag New York, Inc., 2001.

[18] A. Sahai and B. Waters. Ciphertext-policy attribute-based encryption. *Security and Privacy, 2007. SP '07. IEEE Symposium*, pages 321–334, 2007.

[19] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In Ronald Cramer, editor, *Advances in Cryptology EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 457–473. Springer Berlin / Heidelberg, 2005. 10.1007/11426639_27.

[20] P. Samarati and L. Sweeney. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. 1998.

[21] V. Schiavoni, E. Riviere, and P. Felber. Whisper: Middleware for confidential communication in large-scale networks. *Distributed Computing Systems (ICDCS), 2011 31st International Conference*, 2011.

[22] Adi Shamir. Identity-based cryptosystems and signature schemes. *Proceedings of CRYPTO 84 on Advances in cryptology*, 1985.

[23] R. Sherwood, B. Bhattacharjee, and A. Srinivasan. p^5 : A protocol for scalable anonymous communication. In *Security and Privacy, 2002. Proceedings. 2002 IEEE Symposium on*, pages 58–70. IEEE, 2002.

[24] Robert E. Strom, Guruduth Banavar, Tushar Deepak Chandra, Marc Kaplan, Kevan Miller, Bodhi Mukherjee, Daniel C. Sturman, and Michael Ward. Gryphon: An information flow based approach to message brokering. *CoRR*, cs.DC/9810019, 1998.

[25] David Tam, Reza Azimi, and Hans arno Jacobsen. Building content-based publish/subscribe systems with distributed hash tables. In *In International Workshop On Databases, Information Systems and Peer-to-Peer Computing*, pages 138–152, 2003.

[26] Adnan Tariq, Boris Koldehofe, Gerald Koch, and Kurt Rothermel. Providing probabilistic latency bounds for dynamic publish/subscribe systems. In *Proceedings of the 16th ITG/GI Conference on Kommunikation in Verteilten Systemen (KiVS)*. Springer, 2009.

[27] Muhammad Adnan Tariq, Gerald Georg Koch, Boris Koldehofe, Imran Khan, and Kurt Rothermel. Dynamic publish/subscribe to meet subscriber-defined delay and bandwidth constraints. In *The Sixteenth International Conference on Parallel Computing (Euro-Par)*, 2010.

[28] Muhammad Adnan Tariq, Boris Koldehofe, Ala' Altaweel, and Kurt Rothermel. Providing basic security mechanisms in broker-less publish/subscribe systems. In *Proceedings of the 4th ACM International Conference on Distributed Event-Based Systems (DEBS)*, 2010.

[29] Muhammad Adnan Tariq, Boris Koldehofe, Gerald G. Koch, Imran Khan, and Kurt Rothermel. Meeting subscriber-defined QoS constraints in publish/subscribe systems. *Concurrency and Computation: Practice and Experience*, 2011.

[30] Muhammad Adnan Tariq, Boris Koldehofe, Gerald G. Koch, and Kurt Rothermel. Distributed spectral cluster management: A method for building dynamic publish/subscribe systems. In *Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems (DEBS)*, 2012.

[31] Sasu Tarkoma and Christian Prehofer. Techniques for content subscription anonymity with distributed brokers. 2010.

[32] Li Xiao, Yunhao Liu, Wenjun Gu, Dong Xuan, and Xiaomei Liu. Mutual anonymous overlay multicast. *Journal of Parallel and Distributed Computing*, 66(9):1205 – 1216, 2006. <ce:title>Special Issue: Security in grid and distributed systems</ce:title>.

[33] Yong Yu, Bo Yang, Ying Sun, and Sheng lin Zhu. Identity based signcryption scheme without random oracles. *Computer Standards & Interfaces*, 31(1):56–62, 2009.

Declaration

All the work contained within this thesis, except where otherwise acknowledged, was solely the effort of the author. At no stage was any collaboration entered into with any other party.

(Manuel Bischof)