Access Policy Consolidation for Event Processing Systems

Björn Schilling*, Boris Koldehofe*, Kurt Rothermel* and Umakishore Ramachandran†
*Institute for Parallel and Distributed Systems, Universität Stuttgart
Email: firstname.lastname@ipvs.uni-stuttgart.de
†Georgia Institute of Technology
Email: rama@cc.gatech.edu

***Abstract*—Current event processing systems lack methods to preserve privacy constraints of incoming event streams in a chain of subsequently applied stream operations. This is a problem in large-scale distributed applications like a logistic chain where event processing operators may be spread over multiple security domains. An adversary can infer from legally received outgoing event streams confidential input streams of the event processing system. This paper presents a fine-grained access management for complex event processing. Each incoming event stream can be protected by the specification of an access policy and is enforced by algorithms for access consolidation. The utility of the event processing system is increased by providing and computing in a scalable manner a measure for the obfuscation of event streams. An obfuscation threshold as part of the access policy allows to ignore access requirements and deliver events which have achieved a sufficient high obfuscation level.**

***Keywords*-Event Processing; Security; Access control;**

I. Introduction

In business processes, it is essential to detect inconsistencies or failures early. For example, in manufacturing and logistic processes, items are tracked continuously to detect loss or to reroute them during transport. To answer this need complex event processing (CEP) systems have evolved as a key paradigm for business and industrial applications [1], [2]. CEP systems allow to detect situations by performing operations on event streams which emerge from sensors all over the world, e.g. from packet tracking devices.

While, traditionally event processing systems have applied powerful operators in a central way, the emerging increase of event sources and event consumers have raised the need to reduce the communication load by distributed in-network processing of stream operations [3], [4], [5], [6]. In addition, the collaborative nature of today's economy results in large-scale networks, where different users, companies, or groups exchange events. As a result, event processing networks are heterogeneous in terms of processing capabilities and technologies, consist of differing participants, and are spread across multiple security domains [7], [8]. However, the increasing interoperability of CEP applications raises the question of security [2]. It is not feasible for a central instance to manage access control for the whole network. Instead, every producer of information should be able to control how its produced data can be accessed. For example, a company may restrict certain information to a subset of authorized users (i.e. that are registered in its domain).

Figure 1. Access Control & Event Dependency

Current work in providing security for event-based systems covers already confidentiality of individual event streams and the authorization of network participants [9], [10], [11]. In CEP systems, however, the provider of an event looses control on the distribution of *dependent* event streams. This constitutes a major security problem, allowing an adversary to infer information on confidential ingoing event streams of the CEP system.

As an example consider the logistics process illustrated in Figure 1 where a manufacturer wants to deliver an item to a destination. The shipping company determines a warehouse close to the destination, where the item will be shipped to before it will be delivered to the customer. The logistic process is supported by an event processing system, where operators are hosted in the domain of each party and exchange events including potentially confidential information (e.g. the item's *destination* is transmitted to the shipping company). If now a third party receives events related to the *warehouse*, it may draw conclusions about the original event data (i.e. *destination*), in spite of the manufacturer declaring this information as highly confidential and only providing the shipping company with access rights to it.

The goal of this work is to establish access control that ensures the privacy of information even over multiple processing steps in a multi-domain, large scale CEP system. In particular, our contributions are i) an *access policy inheritance* mechanism to enforce access policies over a chain of dependent operators and ii) a scalable method to measure the *obfuscation* imposed by operators on information exchanged in event streams. This allows to define as part of the *access policy* an obfuscation threshold to indicate when the event processing systems can ignore access restrictions,

Published in the Proceedings of the Conference on Networked Systems (NetSys), March 11-15, 2013, Stuttgart, Germany.

http://dx.doi.org/XX.XXXX/NetSys.2013.47

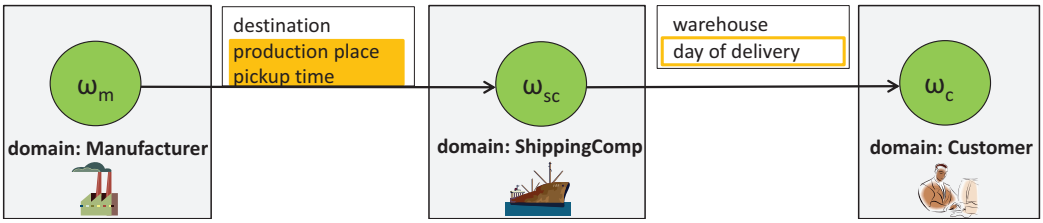


Figure 2. Attributes in Shipping Scenario

thus increasing the number of events to which application components can react to and this way increasing also the utility of the CEP system.

In the remainder of the paper we define the system model and security goal in Section II and Section III respectively. Section IV presents the general concept to establish policy consolidation respecting obfuscation of information. In Section V we enhance the general concept by a local policy consolidation mechanism that overcomes the limitations regarding the scalability of the approach. The evaluation results on the overhead of measuring obfuscation are detailed in Section VI. Finally, we discuss related work in Section VII and conclude our work in Section VIII.

II. System Model

We assume a distributed correlation network, where dedicated hosts are interconnected. On these hosts we deploy operators, which are executed to collaboratively detect situations and form the distributed CEP system. The cooperative behavior of the operators is modeled by a directed operator graph $G = (\Omega, S)$ which consists of operators $\omega \in \Omega$ and *event streams* $(\omega_i, \omega_j) \in S \subseteq (\Omega \times \Omega)$ directed from ω_i to ω_j. Thus, we call ω_i the event *producer* and ω_j the *consumer* of these events. Each event contains one or more event attributes which have discrete values. Every operator ω implements a correlation function $f_\omega : I_\omega \rightarrow O_\omega$ that maps incoming event streams I_ω to outgoing event streams O_ω. In particular, f_ω identifies which events of its incoming streams are selected, how event patterns are identified (correlated) between events, and finally how events for its outgoing streams are produced.

Figure 2 illustrates an operator graph of three operators according to the introduced logistics example, each operator hosted in a distinct domain. The correlation function f_{sc} is applied to events received from and produced by ω_m on produced items in the manufacturing domain. Events produced by f_{sc} carry two *event attributes*, the warehouse location and estimated day of delivery for shipped items.

III. Access Control for CEP

Our approach allows to inherit access requirements by assigning them to event attributes in form of an *access policy*. This allows to preserve requirements through any chain of dependent correlation steps of operators in G. In addition, an obfuscation policy allows to specify an *obfuscation threshold* for event attributes. In each correlation step, the obfuscation of event attributes in produced events is determined by the proposed access policy consolidation protocol. Once the obfuscation threshold is reached for an event attribute, the attribute's access requirements can be ignored. In the following, we detail the concepts behind access policies and obfuscation policies, and formalize the security goal.

A. Access Policies

Access control allows to specify access rights of subjects (operators) for the set of available objects (event attributes). These access rights are provided by the owner of an object (e.g. the producer of an event stream) and may be granted to operators based on an *access requirement*. Such a requirement may be a role, a location or a domain affiliation. Requirements are usually not direct *properties* of the operators, but of the hosts where the operators are deployed. Formally, we specify the access rights within an *access policy* AP for an operator ω as a set of (attribute, access requirement) pairs:

$$AP_\omega = \{(att_1, ar_1), ..., (att_n, ar_n)\}\,.$$

If there is no requirement specified for an attribute, any consumer in the network will be able to access it. Note that we consider attributes to be distinct even if they use the same name, but are produced at two distinct operators.

An access requirement is a tuple of a property p, a mathematical operator op and a value set val: $ar = (p, op, val)$, where $op \in \{=, <, >, \leq, \geq, \in\}$. val can be specified by a range or a set of values. For the sake of simplicity, in this paper access requirements are only referring to domain affiliation and have a structure like this:

$$ar_1 = (domain, \in, \{domainA, domainB\}).$$

In our example scenario, the manufacturer's event attributes have different access requirements. While the information about the item's destination is accessible by the customer, information about where the item is produced and when it can be picked up is restricted to the shipping company. Therefore, the attached AP is defined as follows:

$AP_{manufacturer} =$
{(destination,(domain,∈,{shippingComp,customer})),

(pickup time, (domain,=,shippingComp)),
(production place, (domain,=,shippingComp))}

With the enforcement and assurance of access policies at each producer, a consumer will be eligible to access (receive) an attribute only if the consumer's properties match the access requirements defined for the particular attribute. In this case the consumer is trusted to use the attribute in its correlation function and adopt the requirements specified for the attribute in its own access policy for all produced events.

B. Obfuscation of Event Information

While access policies allow a producer to specify access requirements in a fine-grained manner, the inheritance of requirements in a chain of succeeding operators is at times very restrictive and can limit the efficiency and applicability of the CEP system: in each correlation step of this chain, the number of access requirements may increase by the consolidation of requirements from multiple producers. Each consolidation step can therefore increase the number of interested consumers which are prevented from access to the event attributes of produced event streams. This does not reflect the nature of event processing systems where basic events like single sensor readings may have only little influence on the outcome contained in a complex event representing a specific situation.

In our logistics example, f_{sc} uses *destination*, *production place* and *pickup time* to determine the estimated day of delivery. As a consequence, the customer has no access to the *estimated day of delivery* of the ordered item, since she does not fulfill the access requirements for *production place* and *pickup time*. Yet she has a reasonable interest in this information. And one may claim, that knowledge of the day of delivery does not necessarily allow to draw a relevant conclusion on the *production place* and *pickup time* attribute values. We say, the attribute values get *obfuscated* during the correlation process and depending on the achieved level of obfuscation, the access requirements of an attribute may no longer be needed. In our approach, the level of obfuscation is a measure, to which extent a consumer of the produced attribute (*estimated day of delivery*) can infer the value of the original attribute (*production place*). It can be easily seen in the example, that obfuscation is not only dependent on the values of the attributes, but also on the knowledge of the consumer. Since the *destination* value has led to the *day of delivery* as well, knowledge of the destination would be of great help when trying to infer the restricted attribute *production place* because the delivery time of the item is probably related to the distance between destination and production place. In this work, we will use $obf(att_{old}, att_{new}, \omega)$ to refer to the obfuscation achieved by att_{new} for att_{old} given the knowledge available at a consumer $\omega \in \Omega$.

We allow every operator to specify with its access policy also an obfuscation policy. The obfuscation policy contains obfuscation thresholds for the attributes the operator produces. During the processing of an event attribute, its obfuscation w.r.t. each potential consumer is calculated. Once, the obfuscation threshold for a consumer is reached, the event attribute can be delivered in spite of conflicting access requirements. Formally, we define the obfuscation policy OP for an operator ω as a set of (attribute, obfuscation threshold) pairs:

$$OP_\omega = \{(att_1, ot_1), ..(att_n, ot_n)\}.$$

For instance, the obfuscation policy

$$OP_{manufacturer} = \{(\textit{destination}, 0.9)\}.$$

allows the shipping company for events addressed to the consumer to ignore all access rights for *destination* in the access policy of attribute *day of delivery* if $obf(\textit{destination}, \textit{day of delivery}, \omega_C) \geq 0.9$. We detail the exact semantics of the obfuscation value and its measure in Section IV.

C. Security Goal

Let $att_{old} \rightarrow_\omega att_{new}$ denote that

1) at some operator $\omega \in \Omega$, att_{old} is taken as input to the correlation function f_ω, and
2) f_ω produces att_{new} in dependence of att_{old}.

Furthermore, let $att_{old} \rightarrow^* att_{new}$ denote the transitive closure of the dependency relation. For any pair of attributes with $att_{old} \rightarrow^* att_{new}$ we say that att_{new} is *dependent* on att_{old}. Our main goal is to preserve the privacy of event attributes over multiple correlation steps by respecting the dependency relationship between the attributes produced by the CEP system. In particular, access requirements must not be applied solely to the attribute att_{old}, but have to be inherited to all dependent attributes (att_{new}) unless a sufficient obfuscation threshold for att_{new} has been reached.

More formally, given for each attribute att an initial set of access requirements denoted by $AR_{init}(att)$. We require for any policy consolidation algorithm two conditions to be met:

Condition 1. *For all attributes $att \in O_\omega$ produced at ω*

$$AR_{init}(att) \subset AP_\omega. \quad (1)$$

Condition 2. *For all dependent attribute pairs $(att_{old}, att_{new}) \in \rightarrow^*$ with*

1) *ω_i has produced att_{old} with access requirement $AR(att_{old})$ and obfuscation threshold $(att_{old}, x) \in OP_{\omega_i}$,*
2) *att_{new} is produced by ω_j*
3) *att_{new} is consumed by ω_k*

the access requirement in AP_{ω_j} yield

$$AR(att_{old}) \subset AP_{\omega_j} \textit{ if } obf(att_{old}, att_{new}, \omega_k) < x. \quad (2)$$

A policy consolidation algorithm needs to ensure Condition 1 and Condition 2 in the presence of adversaries who

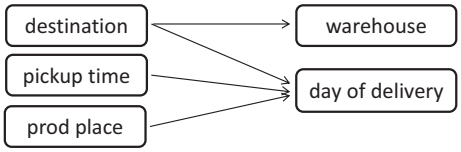


Figure 3. Dependency Graph of the Shipping Company Operator

try to derive event attribute values they are by policy not allowed to access directly.

We want to avoid that hosts maliciously or inadvertently obtain information from event streams for which they have no authorization. Note, by accessing event streams according to the specified system model, hosts may still be able to infer event attributes of unauthorized event streams from legally received event streams. An adversary in our system is therefore limited to the behavior described in the system model. The adversary is authenticated and can only access streams according to its properties. The derived event output follows the operator specification and the access requirements for each executed operator. Each adversary is bound to analyzing outgoing event streams which it is allowed to access, for inferring any additional information.

IV. Policy Consolidation and Event Obfuscation

To meet the security goal from Section III our approach establishes secure event streams between each pair of operators in G. For establishing secure event streams we rely on mechanisms available in state of the art publish/subscribe systems including our own work, e.g. [10], [11], [9], [12], [13]. For our approach it is only important to understand that each consumer ω_c needs to request required event attributes. The requests are handled at the producer ω_p and ω_c will need to authenticate itself against ω_p for the corresponding event attribute. After successful authentication ω_p will forward to ω_c

1) only those events matching the request of ω_c,
2) only those events containing attributes att s.t.
   a) the access policy of att allows ω_c access to att,
   b) att has achieved a sufficiently high obfuscation, i.e. $\forall(att_i, ot_i) \in OP_{\omega_p}\ obf(att_i, att, \omega_c) \geq ot_i$

To this end ω_p will have to perform on its incoming streams an access policy consolidation to ensure all necessary access policies can be inherited and a calculation of the obfuscation values $obf(att_i, att, \omega_c)$. In the following we will show the approach to access consolidation by modeling all potential dependencies between incoming and outgoing event streams in an event dependency graph and calculate obfuscation policies by relying on a Bayesian network.

A. Access Policy Inheritance

Access policy inheritance consists of two basic conceptual steps: First, domain experts have to identify dependencies between incoming and outgoing attributes for each operator. We model these dependencies in a graph as given for our scenario in Figure 3. Second, an operator maps all access requirements specified for each of its incoming attributes to the access policy of all dependent outgoing attributes. In our example scenario, operator ω_{sc} determines the value of the *warehouse* attribute. It needs to map the requirement

$$(domain, \in, \{shippingCompany, customer\})$$

which is associated to the *destination* in $AP_{manufacturer}$ to the *warehouse* attribute, since *warehouse* is dependent on *destination*. Hence, only operators hosted by the shipping company or the customer can access the attribute. After creating the new access policy for the shipping company operator, the access policy contains the following entries:

$AP_{shipping} =$
{(warehouse, (domain,∈,{shippingComp,customer})),
(estArrivalTime, (domain,=,{shippingComp}))}

Since each operator in G will forward only event streams whose event attributes are annotated with consolidated access policies, it is sufficient to consider possible dependencies between event attributes of incoming streams and outgoing streams.

B. Event Obfuscation

While it is easy to model and see dependencies between incoming and outgoing attributes at an operator, it is difficult to have a general purpose measure for the obfuscation of values in event attributes. The level of obfuscation is highly dependent on the correlation function, i.e. how it produces outgoing events based on incoming events. We exemplary show this with two basic operators found in all major CEP systems: a filter, and an aggregator.

A filter's correlation function is simple: for every incoming event it is checked whether one or more attributes have a certain value or are within a certain value range. If so, the events are forwarded to all consumers of the filter operator. Obviously there is no obfuscation of event information and for every received attribute, the consumer can directly infer the values of the original, incoming attributes.

An aggregator is more complex. It collects a set of events within a time window or for a fixed number of events (count) before producing any output. The aggregator combines the attribute values of the incoming events for a newly created output, e.g. the average. As can be seen, the original values from the incoming attributes become obfuscated during the aggregation. The consumers of the aggregated output cannot directly infer the original attribute values. However, depending on the aggregation function one may still guess that the occurrence of some values of incoming attributes is more likely than others. Our goal is to give a general measure for this case.

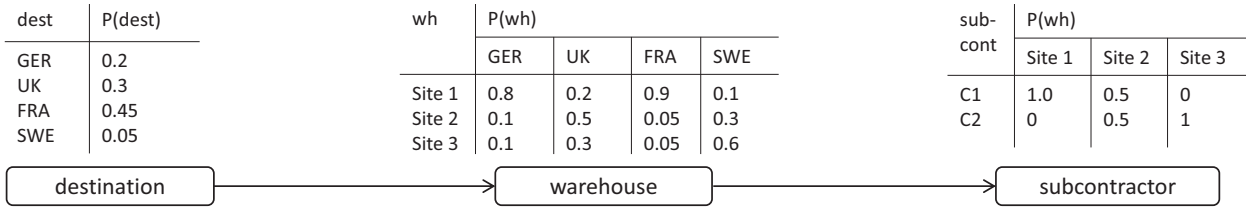


Figure 4. Bayesian Network consisting of Topology and Conditional Probability Tables

Obfuscation semantics: By dealing with obfuscation, our goal is not to influence or control the network participants in how they use the CEP system. In a distributed, heterogeneous CEP system we cannot influence the domain of attribute values, e.g. increasing its size by adding new dummy values. Furthermore, we have no control about the knowledge of operators. This leads to the following strict assumptions we make for obfuscation measurement: We assume a consumer of an attribute att has knowledge about i) the semantics of a correlation function producing att and ii) the possible values of the unknown event attributes att is dependent on.

We measure obfuscation of an unknown event attribute by the equality of likelihood of the different values it can have. This means, if a consumer cannot – based on its knowledge – draw any conclusions on the correct value of an attribute, the attribute is perfectly obfuscated. The obfuscation is perfect if the probability for all possible values of the incoming attribute is equal, i.e. the recipient cannot infer on a value because it was *more likely* to have occurred. Then, maximum obfuscation of 1 is achieved. Consequently, if there is only a single possible value for the incoming attribute, and the user can directly infer on that value, the achieved obfuscation should be 0.

Formalizing Obfuscation: The discussed characteristics lead us to a formalization of attribute inference as a probability value. This inference probability is known as the Bayesian inference and gives us an answer to the question: *Given a certain output attribute, and a certain set of input attributes the consumer knows, how likely is a specific value for the incoming attribute we need to secure*?

We already stated in Section III-B, that the knowledge of the consumer plays an important role when trying to calculate event obfuscation.We model this knowledge as a function $known_{\omega_c}(I_\omega)$, which returns the set of input attributes from I_ω known to the consumer ω_c, i.e. $known_{\omega_c}(I_\omega) \subseteq I_\omega$.

We define that $I^*(att_{new})$ is the set of att_{old} for which $att_{old} \rightarrow^* att_{new}$. Then the inference probability *ip* of an attribute $att_{old} \in I^*(att_{new})$ used to correlate att_{new} for a consumer ω_c is the conditional probability distribution of att_{old}:

$$ip(att_{old}, att_{new}, \omega_c) = P(att_{old} | known_{\omega_c}(I^*(att_{new}) \backslash I^*(att_{old})), att_{new})) \quad (3)$$

As one can see, the inference probability is not a single probability value, but a probability distribution over the value set of the inferred event attribute att_{old}. Furthermore, ip is not only dependent on the operator function, but also on the knowledge of the consumer. Based on the inference probability, we can now measure the obfuscation value achieved for the incoming attribute att_{old} with respect to the outgoing event att_{new} by calculating the entropy of the inference probability distribution:

$$obf(att_{old}, att_{new}, \omega_c) = H(ip(att_{old}, att_{new}, \omega_c)) \quad (4)$$

The entropy gives the desired measure for obfuscation. If the probability for all possible values of the incoming attribute is equal, the entropy of the distribution is 1. Consequently, if there is only a single possible value for the incoming attribute, the entropy and therefore the achieved obfuscation is 0.

Measuring Obfuscation: To measure the obfuscation between two attributes a Bayesian Network is used, since it answers probabilistic queries about the attribute inference [14]. Before being able to query the Bayesian network, it needs to be trained by observing the in- and outgoing events. Every event attribute represents a variable (i.e. node) in the Bayesian Network and every dependency between attributes represents an edge. However, in addition to the dependencies of event attributes, every Bayesian Network associates a probability function with an event attribute. The training algorithm checks, which particular attributes were used by the creation of another attribute. Based on these observations, probability tables are created for every event attribute (cf. Figure 4).

Once the Bayesian network is trained, it can be *queried* about the inference probability of certain attributes. Querying means to provide information about some known event attributes and to calculate the conditional probability distribution of the unknown event attributes. Fitting to our needs, we query the network like this: By providing information about the observed attribute outcome, we receive the probability distribution of the attribute values that have led to the observed outcome. In particular, we can query the inference probability $ip(att_{old}, att_{new}, \omega_c)$ by telling the Bayesian network the observed values for $known_{\omega_c}(I^*_\omega)$ and att_{new}.

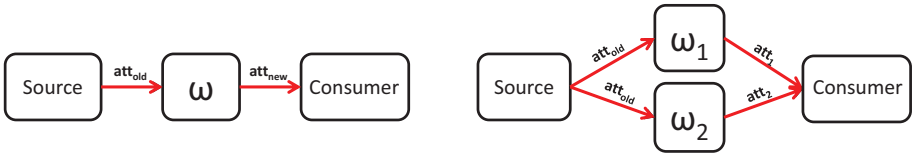


Figure 5. Single- and multiply-connected correlation networks

Scalability Analysis: While the introduced mechanisms fulfill our security goals, the naive application of Bayesian networks does not allow for a scalable introduction for CEP. Calculating the inference probability is NP-hard, adding a potentially infeasible amount of latency to the event processing, if the size of the Bayesian network is big.

To meet the large computational effort of calculating Bayesian inference, two different types of optimizations exist. On the one hand, sampling techniques can be used to estimate the conditional probabilities of the Bayesian network (e.g. [15], [16]). However, their precision depends strongly on the number of samples taken from the network, and no approximation scheme exists that allows to draw samples in polynomial time to achieve a certain precision. This makes the approximate algorithms infeasible for security applications, since no guarantees can be made in appropriate time [14]. On the other hand the complexity of calculating *exact* inference can be reduced by storing partial results of the inference calculation which otherwise would have to be calculated multiple times (e.g. [17]). However, the benefit of these optimizations is heavily dependent on the structure of the Bayesian network. They work well in simple, single connected Bayesian networks (cf. Figure 5) where the time-complexity can be reduced to be linear to the number of attribute values.

Besides the computational effort, creating and querying a Bayesian network also requires communication, if the needed information is spread among a network. Also, in order to calculate obfuscation over a chain of operators, additional communication will be needed, since the inference probability relies on the consumer's *knowledge* (cf. Equation 5).

As can be seen, the time and communication needed to calculate the inference probability can be huge because all dependent event attributes need to be considered ($I^*(att_{new})$). Hence, calculating the inference for the transitive closure of dependable attributes is not scalable. We tackle this by dividing the problem, and therefore the Bayesian Network, in multiple parts that allow to be treated independently.

V. Scalable Access Policy Consolidation

Instead of accounting for a global Bayesian network, we propose to exploit local knowledge available at each host. This allows us to reduce the number of relations of incoming and outgoing attributes and thus leads to a huge gain in processing overhead. The idea of our approach is that a host in the CEP network creates a local Bayesian network for each of its deployed operators. The handling (i.e. forwarding) of the event is based on the locally achieved obfuscation. This limits the computational effort by accepting that obfuscation is not measured over multiple correlation steps, and therefore some events may be treated more restrictive than actually needed.

Algorithm 1 Local Obfuscation Calculation

```
procedure INITIALIZE(ω)
    for all operator ω do
        D_ω ← FINDMULTIPATHOPERATORS(ω)
    end for
    for all ω ∈ D_ω do
        relAtts ←FINDRELATEDATTRIBUTES
        for all (att_new, att_old) ∈ relAtts do
            TRANSMIT P(att_new|att_old) TO ω
        end for
    end for
end procedure

procedure UPONRECEIVEEVENT(e)
    for all att ∈ e do
        if ∃ multPathDependency(att) then
            CALCULATEWORSTCASEOBFUSCATION(ATT)
        else
            CALCULATELOCALOBFUSCATION(ATT)
        end if
    end for
end procedure
```

A. Measuring Local Obfuscation

In the approach, every host calculates obfuscation only for the locally known attribute dependencies (i.e. $att_{old} \rightarrow_\omega att_{new}$) in contrast to calculating the obfuscation for every pair of dependent attributes (i.e. $att_{old} \rightarrow^* att_{new}$). This has three major benefits: i) a smaller dependency graph, ii) less communication overhead, and iii) the network is not multiply connected, because there exist only paths of length 1. As a consequence, every host can create a *local dependency graph* on its own instead of creating a global dependency graph for all dependent attributes. Furthermore, we can efficiently calculate the exact inference probabilities by applying variable elimination optimization for single connected networks to efficiently determine the obfuscation value (cf. Section IV-B).

Even in a local approach for obfuscation calculation the multi-path dependencies of attributes need to be considered. Attributes might reach the recipient via multiple paths (i.e. parallel chains of operators in a multiply-connected correlation network, cf. Figure 5). An adversary that can subscribe to such attributes may be able to infer the original value by combining the event information received through the multiple paths. We meet this by analyzing the entire operator graph during initialization of our algorithm (c.f. Algorithm 1). For every attribute pair with multi-path dependencies the operators that reside on distinct paths exchange the dependency functions w.r.t. the attributes. For example, in a scenario as depicted in Figure 5, the inference probability is calculated as follows:

$$\begin{aligned} &P(att_{old}|att_1, att_2) \\ &\quad = \alpha * P(att_{old}) * P(att_1|att_{old}) * P(att_2|att_{old}) \end{aligned} \tag{5}$$

where α is the normalization constant $1/P(att_2)$.

Hence, $P(att_1|att_{old})$ is sent to operator ω_2 and $P(att_2|att_{old})$ to operator ω_1 vice versa.

After performing the initialization, each operator can calculate the obfuscation value from local knowledge only. In the above example, if operator ω_1 now calculates the obfuscation of an incoming attribute att_{old} for the outgoing attribute att_1, it uses the dependency functions received during the initialization phase. There, it searches for the outcome att_2 which has the highest chance for inferring att_{old}, i.e. the entry with the highest probability. This value is then used in the calculation of $P(att_{old}|att_1, att_2)$, as it results in the minimal achievable obfuscation.

Note that the initialization needs to be performed with each change of the correlation graph and follows the learning phase of the Bayesian networks. However, changes to the operator graph typically are for many practical settings a result of changes to the business logic. Hence, we expect only rare interruptions of the event processing service.

B. Correctness

As our work addresses mainly how to establish producer centric access policies in CEP in a scalable way, we give only informal correctness arguments under the limitations for the adversary introduced in Section III. Three main properties guarantee that the proposed approach is correct in terms of the defined security goal:

1) According to our assumptions in Section III, an adversary tries to infer additional information by analyzing all event streams which it is allowed to access. The proposed algorithm considers the complete knowledge the consumer *might* have. That means, it is considered that every attribute influencing the requested local obfuscation ($obf(att_{old}, att_{new}, \omega_c)$) that is accessible to the consumer is known.
2) In accordance to Property 1, every path from att_{old} to att_{new} is considered in the algorithm. That means, every piece of information an adversary may access in order to infer att_{old} is included when calculating the inference.
3) Locally unknown events (which may occur in multi-path dependency calculations) are always handled as a worst-case-consideration. We always use the value in our calculations which would give an adversary the most inference information, i.e. the value resulting in the worst obfuscation.

Since all sources of event information which might influence the obfuscation value of any operator are considered in our approach, the obfuscation value calculated at an operator cannot further be lowered by any means. Hence, with the presented approach, we guarantee: If the consumer does not fulfill the access requirements for an attribute att_{old}, it will also not be able to access any attribute att_{new} if the attributes depend on each other ($(att_{old} \rightarrow^* att_{new})$ unless a sufficient obfuscation threshold for att_{new} has been reached. We do not guarantee, though, that the consumer will receive every attribute that has achieved a sufficient obfuscation.

VI. Discussion and Evaluation

We implemented the presented approach within the DHEP framework [7] which enables CEP in a heterogeneous environment. That means, hosts may be spread among different security domains and have differing processing capabilities or use different correlation engines. Hence, using the framework allows us to create multi-domain distributed CEP networks.

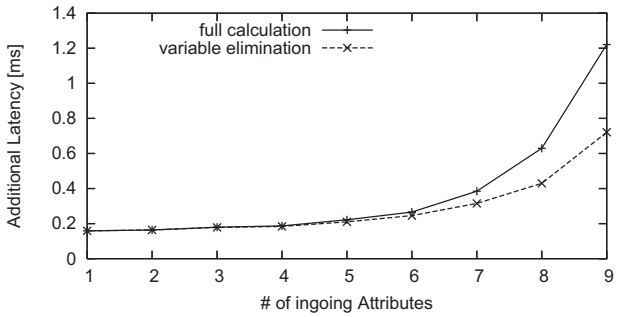
To achieve policy consolidation, every operator receiving a request provides the requester with the information needed for further processing: the access policy as well as the obfuscation policy. The policies might be different depending on the consumer (see Section IV). The events a consumer receives as well as its adherence to access policy inheritance is dependent on whether it fulfills the access requirements. To realize the obfuscation measurement we make use of the Weka framework [18]. Weka is a data mining tool which comes with a Bayesian network implementation. Furthermore, it allows us to add hidden variables which is needed to compute multi-path inference as discussed in Section V. Every host in our framework runs its own implementation of Weka. For every event attribute produced, we calculate the achieved obfuscation and forward it to potential recipients in accordance to the obfuscation. Weka does not provide any optimization for calculating the Bayesian inference. Instead, it uses the naive full calculation. To measure the computational effort of the *variable elimination* optimization (see Section IV-B), we provide an own algorithm implementation.

Access policy consolidation reduces the network usage. This is due to the fact that both number and size of events

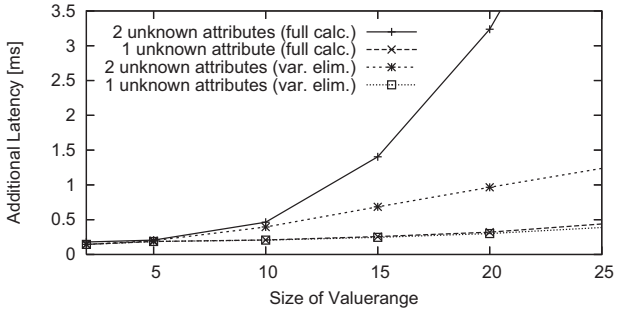
decrease because not all events or event attributes will be received by an operator. However, it can be easily seen that this reduction is fully dependent on the application characteristics, especially on the access rights of the operators and the frequency distribution of event attribute values. Therefore it is not possible to provide meaningful evaluations and we focus on evaluations of the additional latency caused by our approach.

Despite reducing the network usage, policy consolidation will also cause additional latency for event processing on the network nodes. Although we can reduce the computational effort by only considering local obfuscation, the computation still takes a considerable amount of time. The computational effort is mainly dependent on the size of the Bayesian Network and the number of consumers, since different consumers can have differing obfuscation. We analyzed the additional latency caused by our policy consolidation mechanism both dependent on the number of input attributes as well as the number of attribute values. We used a simple setup, where one operator receives events containing one attribute. In our evaluations, both the size of the attribute domain as well as the number of event sources vary. The operator is hosted on a machine with a 2GHz CPU and 3GB main memory, where the introduced Weka framework (as well as our external optimization algorithm) is deployed. The incoming events are processed by an ESPER correlation engine which creates an output event, containing one attribute, once events from all sources are received. For the new created event, we calculate the achieved obfuscation for a consumer. To have results independent of the processing time of the used correlation engine, we extracted and depicted only the time needed for calculating inference in the Bayesian Network, since it is the main source for additional latency caused in our approach.

Figure 6(a) depicts the additional latency depending on the number of event sources. The number of event sources has a direct influence on the size of the locally created dependency graph, hence on the size of the Bayesian Network. No ingoing attribute was known to the consumer. The size of the attribute domain was fixed to two, meaning that every event attribute was boolean. The results show that the increase of the latency, caused by the computation of obfuscation values increases exponentially with the total number of attributes. This behavior is expected (c.f. Section IV-B). However, computations are fast for networks with a small number of attributes, as they are common in many CEP applications. Since security-related event systems have, depending on the network and event parameters, a processing time in the range of one millisecond and more per event [19], [20], [9], we consider a latency of up to 1ms as acceptable for our approach. In our second evaluation, we leave the number of event sources fixed at two but varied the domain size (cf. Figure 6(b)). Furthermore, we calculate the achieved obfuscation for two different consumers. One consumer has



(a) Latency vs. Attributes



(b) Latency vs. Size of Domain

Figure 6. Measuring additional Latency

no knowledge about any ingoing attribute, while the other has knowledge about one ingoing attribute. We calculate the obfuscation for the other event attribute, which both consumers might try to infer. As can be seen, the optimized algorithm proves to be significantly faster than the standard calculations, if there is more than one unknown event source. It can be seen that variable elimination reduces the complexity to be linear dependent on the size of the attribute domain, and our approach benefits heavily from it.

Our results show that the additional processing time is highly dependent on the number of unknown attributes in the dependency graph as well as the number of potential values each of the unknown attributes might have. It can be seen, that the size of the attribute domain is less critical than the number of attributes. This fits well with many CEP systems, where it is unusual to correlate events from many different sources, but rather have a limited number of sources with potentially large attribute value ranges. We conclude that obfuscation calculation is reasonable if the application characteristics allow for it. The calculation may not be feasible for applications with very high event rates, since measuring the obfuscation would take too long and the processing of events would be slowed down. One solution for this kind of applications may be to calculate a static, worst case obfuscation instead of calculating the obfuscation for every new event.

VII. Related Work

With the increasing popularity of event-driven systems, a lot of effort has been spent to make the systems secure. For example, a role-based access control is proposed in [3]. Pesonen et al. and Bacon et al. discuss how publish/subscribe systems can be secured by introducing access control policies in a multi-domain architecture [10], [11]. They describe how event communication between the domains can be supported. Opyrchal et al. present the concept of event owners that can be specified. These are used to provide access to *their* events [21]. Tariq et al. propose a solution to provide authentication and confidentiality in broker-less content-based publish/subscribe systems [9]. Our work is based on the previous work which make event communication secure among different entities in the system. We assume the presence of a system that can handle access control on events. Based on this, we use policy composition in order to derive the necessary access policies at any point during the event processing steps.

Access policy composition has found a lot of consideration in distributed systems. Bonatti et al. defined a well recognized algebra for composing access policies [22]. Especially in the area of web service composition, the composition of security policies plays an important role, as different policies have to be combined for every combination of web services (e.g. [23], [24]). We adopt some of these concepts into our distributed CEP system, which allows us to inherit access restrictions during the different processing steps in the operators of our system.

To realize our concepts we make use of techniques from statistical inference. More specific, we calculate the Bayesian inference after creating a Bayesian network and learning the dependencies (e.g. [14], [18]). Since Bayesian inference is a complex calculation, several Monte-Carlo algorithms have been proposed to estimate the inference value(s). They all have in common to arbitrarily pick samples from the Bayesian network probability distribution, and estimate the values based on the samples. The precision of the estimated inference values is dependent on the number of samples. A commonly used technique is the Gibbs sampler [15], [16].

VIII. Conclusion

This paper addressed the inheritance and consolidation of access policies in heterogeneous CEP systems. We identified a lack of security in multi-hop event processing networks and proposed a solution to close this gap. More specific, we presented an approach that allows the inheritance of access requirements, when events are correlated to complex events. Our algorithm includes the obfuscation of information, which can happen during the correlation process, and uses the obfuscation value as a decision-making basis whether inheritance is needed. We presented an implementation of our approach, based on Bayesian Network calculations. The analysis and evaluations show that the approach is computation-intensive, once the Bayesian Network grows, hence raising the processing time of an event. To deal with the calculation cost, we introduced a local approach, where every participant calculates local obfuscation achieved during the correlation process. We use a variable elimination optimization to further reduce the computational effort for calculating obfuscation. Future work will concentrate on enhancing the obfuscation calculation and methods to increase the Bayesian Network size so we are able to measure obfuscation over more than one correlation steps.

Acknowledgments

This work is supported by contract research "CEP in the Large" of the Baden-Württemberg Stiftung. The authors would like to thank G. Koch, A. Grau, and the reviewers for their helpful comments.

References

[1] A. Buchmann and B. Koldehofe, "Complex event processing," *it - Information Technology*, vol. 51:5, pp. 241–242, 2009.

[2] A. Hinze, K. Sachs, and A. Buchmann, "Event-based applications and enabling technologies," in *Proceedings of the Third ACM International Conference on Distributed Event-Based Systems*, ser. DEBS '09. New York, NY, USA: ACM, 2009, pp. 1:1–1:15.

[3] P. Pietzuch, "Hermes: A scalable event-based middleware," Ph.D. dissertation, University of Cambridge, 2004.

[4] G. Li and H.-A. Jacobsen, "Composite subscriptions in content-based publish/subscribe systems," in *Proc of the 6th Int. Middleware Conf.*, 2005, pp. 249–269.

[5] G. G. Koch, B. Koldehofe, and K. Rothermel, "Cordies: expressive event correlation in distributed systems," in *Proc. of the 4th ACM International Conference on Distributed Event-Based Systems (DEBS)*, 2010, pp. 26–37.

[6] B. Koldehofe, B. Ottenwälder, K. Rothermel, and U. Ramachandran, "Moving range queries in distributed complex event processing," in *Proc. of the 6th ACM International Conference on Distributed Event-Based Systems (DEBS)*, 2012, pp. 201–212.

[7] B. Schilling, B. Koldehofe, U. Pletat, and K. Rothermel, "Distributed heterogeneous event processing: Enhancing scalability and interoperability of CEP in an industrial context," in *Proc. of the 4th ACM International Conference on Distributed Event-Based Systems (DEBS)*, 2010, pp. 150–159.

[8] B. Schilling, B. Koldehofe, and K. Rothermel, "Efficient and distributed rule placement in heavy constraint-driven event systems," in *Proc. of the 10th IEEE International Conference on High Performance Computing and Communications (HPCC)*, 2011, pp. 355–364.

[9] M. A. Tariq, B. Koldehofe, A. Altaweel, and K. Rothermel, "Providing basic security mechanisms in broker-less publish/subscribe systems," in *Proceedings of the 4th ACM Int. Conf. on Distributed Event-Based Systems (DEBS)*, 2010, pp. 38–49.

[10] L. I. W. Pesonen, D. M. Eyers, and J. Bacon, "Encryption-enforced access control in dynamic multi-domain publish/subscribe networks," in *Proc. of the 2007 ACM International Conference on Distributed Event-Based Systems (DEBS)*, 2007, pp. 104–115.

[11] J. Bacon, D. M. Eyers, J. Singh, and P. R. Pietzuch, "Access control in publish/subscribe systems," in *Proc. of the 2nd ACM International Conference on Distributed Event-Based Systems (DEBS)*, 2008, pp. 23–34.

[12] M. A. Tariq, B. Koldehofe, G. G. Koch, I. Khan, and K. Rothermel, "Meeting subscriber-defined QoS constraints in publish/subscribe systems," *Concurrency and Computation: Practice and Experience*, vol. 23, no. 17, pp. 2140–2153, 2011.

[13] S. Rizou, F. Dürr, and K. Rothermel, "Providing qos guarantees in large-scale operator networks," in *High Performance Computing and Communications (HPCC), 2010 12th IEEE International Conference on*, 2010, pp. 337 –345.

[14] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach, 2nd ed.* Prentice Hall, 2002.

[15] S. Geman and D. Geman, "Stochastic relaxation, gibbs distributions, and the bayesian restoration of images," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. PAMI-6, pp. 721 –741, 1984.

[16] A. E. Gelfand and A. F. M. Smith, "Sampling-based approaches to calculating marginal densities," *Journal of the American Statistical Association*, vol. 85, no. 410, pp. 398–409, 1990.

[17] N. L. Zhang and D. Poole, "Exploiting causal independence in bayesian network inference," *Journal of Artificial Intelligence Research*, vol. 5, pp. 301–328, 1996.

[18] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: an update," *SIGKDD Explor. Newsl.*, vol. 11, pp. 10–18, November 2009.

[19] M. Srivatsa and L. Liu, "Securing publish-subscribe overlay services with eventguard," in *Proceedings of the 12th ACM conference on Computer and communications security*, 2005, pp. 289 – 298.

[20] C. Raiciu and D. S. Rosenblum, "Enabling confidentiality in content-based publish/subscribe infrastructures," in *Securecomm and Workshops, 2006*, 28 2006-sept. 1 2006, pp. 1 –11.

[21] L. Opyrchal and A. Prakash, "Secure distribution of events in content-based publish subscribe systems," in *In Proceedings of the 10th USENIX Security Symposium*, 2001, pp. 281–295.

[22] P. Bonatti, S. De Capitani di Vimercati, and P. Samarati, "An algebra for composing access control policies," *ACM Trans. Inf. Syst. Secur.*, vol. 5, pp. 1–35, February 2002.

[23] F. Satoh and T. Tokuda, "Security policy composition for composite web services," *Services Computing, IEEE Transactions on*, vol. 4, pp. 314 –327, 2011.

[24] A. J. Lee, J. P. Boyer, L. E. Olson, and C. A. Gunter, "Defeasible security policy composition for web services," in *Proc. of the 4th ACM Workshop on Formal Methods in Security*, 2006, pp. 45–54.