Selective Access for Supply Chain Management in the Cloud

Anselme Tueno*, Florian Kerschbaum†, Daniel Bernau* Sara Foresti‡,
*SAP Research, Karlsruhe, Germany – *Email*: anselme.kemgne.tueno@sap.com, daniel.bernau@sap.com
†University of Waterloo, Waterloo, Canada – *Email*: florian.kerschbaum@uwaterloo.ca
‡Università degli Studi di Milano, 26013 Crema, Italy – *Email*: sara.foresti@unimi.it

***Abstract*—Object-level tracking along supply chains, enabled by the low-cost and wide availability of Radio Frequency Identification (RFID) technology, permits companies to collect large amounts of data (e.g., time, location, handling) about the goods they produce. Combining the data collected by the different companies along a supply chain can provide considerable advantages to all of them. However, such a sharing sometimes needs to be selective. Indeed, companies may need to keep some information about their business operations secret. In this paper, we propose a solution that enables selective sharing of data collected along a supply chain. Our solution uses the services offered by cloud providers for sharing data among companies, and relies on selective encryption for enforcing access restrictions over such data.**

***Index Terms*—Selective encryption, supply chain policy**

I. Introduction

Thanks to the wide availability today of Radio Frequency Identification (RFID) technology [13] at limited prices, companies are more and more adopting object-level tracking in their supply chains. Indeed, object-level tracking provides business benefits and is sometimes requested by regulations (e.g., in the pharmaceutical industry). RFID technology provides the means to equip and capture each object with an unique identifier and hence enables companies to easily collect information about it. Data commonly collected in supply chains include time, location, and type of handling (e.g., packing, unpacking, receiving, or shipping). On one hand, combining these data from all the companies along a supply chain (not just predecessor and successor of each phase) enables or improves many economically attractive collaborative applications, including batch recalls [29], counterfeit detection [28], benchmarking and analytics [18]–[20], or estimated arrival forecasts [6]. On the other hand, information collected along the supply chain may be considered sensitive as it allows espionage on the business operations of the involved companies [12], [23]. The need for keeping a subset of the collected data secret, also to (some of) the other companies along the supply chain, represents a major obstacle to wider sharing of object-level tracking data.

In this paper, we propose a solution for enabling companies collaborating in the production process to selectively share object-level tracking data. The proposed solution leverages the services offered by the cloud for data sharing and relies on a Trusted Third Party (TTP) offering a Public Key Infrastructure (PKI). Therefore, collected data are stored in a centralized cloud database that all the collaborating companies can access. Since the cloud provider itself may not be authorized for the sensitive content of the collected data, and then also to enforce access restrictions, the proposed approach relies on selective encryption for enforcing the authorization policy regulating access to collected data. Intuitively, each company will encrypt different portions of the collected information using different encryption keys, which are then distributed to the other companies along the chain, in such a way that each company can decrypt all and only the data it is authorized to access.

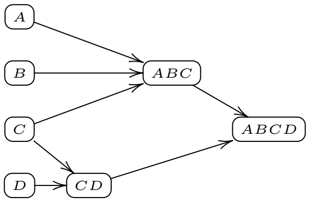
	t_1	t_2	t_3	t_4
A	1	1	0	1
B	1	0	0	1
C	1	0	1	1
D	0	0	1	1

Fig. 1: An example of access matrix

The reminder of this paper is organized as follows. Section II illustrates how selective encryption can be used to enforce access control restrictions. Section III focuses on the policies that should be used to regulate visibility over objects' data in supply chain scenarios. Section IV introduces our approach, based on selective encryption, for effectively enforcing these visibility policies. Section V summarizes related works. Finally, Section VI presents our conclusions.

II. Selective Encryption for Access Control Enforcement

Given a relation r, defined over relation schema $R(a_1,\ldots,a_n)$, the authorization policy regulating access to the tuples in r by the users in the system can be represented through an *access matrix* AM, having a row for each user $u \in \mathbb{U}$ and a column for each tuple $t \in r$. Cell $AM[u,t]$ in the access matrix has value 1 iff u is authorized to access t, it has value 0 otherwise. The set of users who can access a tuple t represents its *access control list*, $acl(t) \subseteq \mathbb{U}$. Figure 1 illustrates an example of access matrix regulating access to a relation composed of 4 tuples $\{t_1,\ldots,t_4\}$ for a set $\mathbb{U}$=$\{A,B,C,D\}$ of 4 users. As an example, $acl(t_1)$=ABC.



(a)

user	key
A	k_A
B	k_B
C	k_C
D	k_D

(b)

tuple	key
t_1	k_{ABC}
t_2	k_A
t_3	k_{CD}
t_4	k_{ABCD}

(c)

source	target	token
l_A	l_{ABC}	$k_{ABC}\oplus h(k_A,l_{ABC})$
l_B	l_{ABC}	$k_{ABC}\oplus h(k_B,l_{ABC})$
l_C	l_{ABC}	$k_{ABC}\oplus h(k_C,l_{ABC})$
l_C	l_{CD}	$k_{CD}\oplus h(k_C,l_{CD})$
l_D	l_{CD}	$k_{CD}\oplus h(k_D,l_{CD})$
l_{CD}	l_{ABCD}	$k_{ABCD}\oplus h(k_{CD},l_{ABCD})$
l_{ABC}	l_{ABCD}	$k_{ABCD}\oplus h(k_{ABC},l_{ABCD})$

(d)

Fig. 2: Encryption policy equivalent to the authorization policy in Figure 1

Selective encryption represents an effective solution for enforcing access control restrictions over data stored at an external, possibly not fully trusted, cloud provider. Indeed, selective encryption guarantees that the data self-enforce access control restrictions. Hence, the enforcement of the authorization policy regulating access to the data requires the intervention neither of the data owner nor of the cloud provider. Intuitively, selective encryption consists in encrypting different tuples with different encryption keys, and in distributing keys in such a way that the key used to protect a tuple is known to all and only the users authorized to it. The adoption of selective encryption requires the definition of an *encryption policy* equivalent to AM. The encryption policy establishes the key(s) to be used to encrypt each tuple $t \in r$ as well as key distribution to users. An encryption policy is equivalent to, and hence correctly enforces, an authorization policy if each user is able to decrypt all and only the tuples she is authorized to access (i.e., u can decrypt t iff $AM[u,t]$=1).

To translate the authorization policy into an equivalent encryption policy, the solution in [10], [11] defines a different key k_u for each user $u \in \mathbb{U}$ and a key k_U for each set U of users representing the access control list of a tuple $t \in r$. Each user u is then assigned key k_u and each tuple $t \in r$ is encrypted with k_U, with $U = acl(t)$. Equivalence between the encryption policy and the authorization policy is guaranteed by using a *token-based key derivation* technique [1], enabling the derivation of each key k_U from any key k_u such that $u \in U$. For key derivation, each encryption key k_i is associated with a public label l_i. Token $token_{i,j}$=$k_j \oplus h(k_i,l_j)$, with $\oplus$ the bitwise xor operator and h a collusion resistant cryptographic hash function, permits to derive key k_j from the knowledge of k_i and public label l_j. Graphically, keys and tokens can be represented in a *key derivation graph* with a node for each key k and an edge (k_i,k_j) for each token $token_{i,j}$. Equivalence to the authorization policy is then guaranteed if there exists a path from k_u to k_U iff $u \in U$. Indeed, this permits any user u authorized to access a tuple t to derive key k_U, with $U = acl(t)$, used to encrypt t starting from the knowledge of her own key k_u. Figure 2 illustrates an encryption policy equivalent to the authorization policy in Figure 1. Figure 2(a) illustrates the key and token graph, while Figures 2(b-c) summarize the keys assigned to users and the keys used to encrypt tuples, respectively. Figure 2(d) reports publicly available tokens, stored at the cloud provider.

Since the encryption policy should be equivalent to the authorization policy to guarantee its proper enforcement, updates to the authorization policy translate into updates to the corresponding encryption policy. Intuitively, every time a user is granted (revoked, respectively) access to a tuple, it should be (re-)encrypted with a key known to all and only the users in the new access control list. If such a key does not exist, the data owner needs to generate it, together with the set of tokens necessary to enable its derivation. For instance, if user B is granted access to t_2, then $acl(t_2)$=AB. To enforce such a policy update, the data owner needs to generate a new encryption key, k_{AB}, and the tokens that enable users A and B to derive it. She will then re-encrypt t_2 with k_{AB}. Note that, in some situations, it is not necessary to generate a new key and re-encryption can be avoided. If user u is granted access to a tuple t and no other tuple is encrypted with the same key as t, it is sufficient to add a token enabling u to derive the key used for t. For instance, if user A is granted access to t_3, then $acl(t_3)$ becomes ACD. Since k_{CD} is used to protect t_3 only, the data owner can simply add a token enabling A to derive k_{CD} from k_A.

III. Supply Chain

In this section, we illustrate the problem facing supply chain partners, and describe visibility policies and the access control protocol that will allow to enforce fine-grained access to supply chain data.

A. Problem Definition

Imagine a set of mobile physical objects $o_1, ..., o_m$ each traversing a (potentially different) subset of players $p_1, ..., p_n$. Each player p_i collects information about each object o_j it handles (e.g., time, place, type of action, etc.) and stores this information in a central cloud database. Later other players may ask to access an object's data in this database. This scenario is common place in modern supply chains [27]. Figure 3 illustrates an example of a supply chain where a manufacturer p_1 produces two objects o_1, o_2, collects information I_{11}, I_{21} and ships both objects to distributor p_2. The distributor collects information I_{12}, I_{22} and ships o_1, o_2 to retailers p_3, p_4 respectively. Later, a supply chain partner may want to access the data collected by another one.

Companies are usually part of many supply chains (even for the same product), and collected data may be stored in a single cloud database. Specifying access control rules for the tuples in this database can be very delicate. Consider the following two examples. Imagine a supplier p_2 selling a product o_2 to

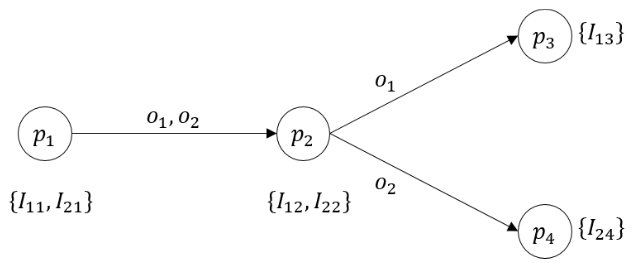


Fig. 3: An example of a supply chain

buyers p_3 and p_4. If buyer p_3 has access to all scheduled orders for o_2, she can infer the volume of future business with p_4. This can be very sensitive, in case supplier p_2 has to cancel some orders due to a temporary capacity reduction (e.g., a machine failure). Buyer p_3 could then infer whether p_4's orders are treated preferentially. While this decision can be based on local information, in the case of bridging only one supply chain stage, it becomes difficult in case of a tier-2 supplier (i.e., a supplier's supplier). As another example, imagine a supplier p_1 selling product o_1 to p_2 which is then used by p_2 to produce o_2. Again, assume that p_2 later sells object o_2 to players p_3 and p_4. If either buyer p_3 or p_4 contacts supplier p_1 requesting data, p_1 cannot decide which object was shipped to which buyer. If supplier p_1 would grant access to all items, buyer p_3 could infer again the volume of business of p_4.

A naturally emerging access rule is to share data with partners about shared objects, that is, objects a group of partners have possessed. This implements the important business concept of visibility [21], that is, each partner gains (additional) information about how its (entire) supplies are produced and how its (entire) products are used, but still provides a separation between different supply chains merging at one company. Furthermore it can be easily adopted reciprocally (i.e. "I give you access, if you give me access") providing a fair allocation of cost, risk, and benefits.

We distinguish between *downstream* and *upstream* visibility. In downstream visibility a company is allowed to access data associated with its objects shipped to its supply chain partners (at those partners). Upstream visibility is the reverse and a company is allowed to access data associated with objects it has received from its supply chain partners (again at those partners).

B. Visibility Policies

Visibility policies allow each partner to gain information about how its supplies are produced and how its products are used, but still provides a separation between different supply chains merging at one company. We now review the notion of visibility policies introduced in [21]. To this purpose, we first introduce the trajectory of an object o_j, and then present the definition of upstream and downstream visibility policies.

Definition 1: Let there be n players $p_i \in \mathbb{X} = \{p_1, ..., p_n\}$. The *trajectory* $L(o_j) = \langle \mathbb{L}_j, \mathbb{R}_j \rangle$ of object o_j is modeled as a totally ordered set consisting of elements in the set $\mathbb{L}_j \subseteq \mathbb{X}$ and a binary relation $\mathbb{R}_j \subseteq \mathbb{L}_j \times \mathbb{L}_j$. The players represent the spatial domain of the trajectory and the players in $\mathbb{L}_j$ are those that have handled (possessed) the object o_j. Relation $\mathbb{R}_j$ models the temporal domain of the trajectory.
Simply speaking, a player p_i is ranked lower than a player $p_{i'}$ in $L(o_j)$, that is, $\langle p_i, p_{i'} \rangle \in \mathbb{R}_j$, if p_i handled the object o_j earlier than $p_{i'}$. We write $\sigma(p_i, L(o_j))$ for a *predicate* that can be used to compute the *rank* of player p_i in $L(o_j)$ and $|\sigma(p_i, L(o_j))|$ for the evaluated rank itself. The predicate $\sigma(p_i, L(o_j))$ can be seen as a chained proof of the path followed by object o_j to player p_i. This means, each sender adds a proof to the chain that it sent the object to the next receiver. In this case, we see $|\sigma(p_i, L(o_j))|$ as the length of the proof $\sigma(p_i, L(o_j))$ if it is indeed true. We also assume the existence of an external party, the trusted third party, that helps to prove the source of the chain and to compute the rank (Section III-C). Then, $|\sigma(p_i, L(o_j))| < |\sigma(p_{i'}, L(o_j))|$ iff $\langle p_i, p_{i'} \rangle \in \mathbb{R}_j$. If player p_i did not handle the object o_j, then $|\sigma(p_i, L(o_j))|$ is UNDEFINED and any order relation on the natural numbers, (e.g., both $<$ and $>$), should always evaluate to false. We say the least element in $L(o_j)$ is the *source* of object o_j and the top most element is its *destination*.

In the example in Figure 3, the trajectory of o_1 is $L(o_1) = \langle \{p_1, p_2, p_3\}, \{(p_1, p_2), (p_2, p_3)\} \rangle$. As the proof of the source is provided by the trusted third party, the rank of p_1, p_2, p_3, p_4 are respectively $1, 2, 3,$ UNDEFINED.

A player p_i is part of multiple supply chains if at least two objects that it handled have been handled by at least one player each - both upstream or downstream - which did not handle both objects. For example distributor p_2 is in the supply chain of both objects o_1 and o_2.

Now assume, player p_i is requesting information from player p_v (verifier) about object o_j stored in the cloud database. Player p_v intercepts this request and performs an access control decision. The intercepting component is called a policy enforcement point (PEP) and the information about the request, (e.g., the identity of the requestor p_i and the unique identifier of the object o_j) are forwarded to the policy decision point (PDP). The PDP compares the information with the policies in its store and returns its evaluation decision (grant or deny) to the PEP, which will then enforce grant (or deny) access.

Definition 2: An *upstream visibility policy* grants (or denies) access to p_i for o_j based on the predicate evaluation

$$|\sigma(p_v, L(o_j))| < |\sigma(p_i, L(o_j)|.$$

With reference to the example in Figure 3, player p_3 can be granted access to the information collected at p_1,p_2 only on object o_1. Similarly, player p_4 can be granted access to information collected at p_1,p_2 only on object o_2.

Definition 3: A *downstream visibility policy* grants (or denies) access to p_i for o_j based on the predicate evaluation

$$|\sigma(p_i, L(o_j))| < |\sigma(p_v, L(o_j)|.$$

With reference to the example in Figure 3, player p_1 can be granted access to the information collected at p_2, p_3, p_4 on objects o_1 and o_2.

C. Access Control

Existing access control models are not scalable when protecting object-level data, because the authorization matrix is too huge. Since the concept of visibility is independent of the players on the trajectory of an object, it can be used to reduce the authorization matrix and simplify its administration. Moreover, it is possible to unify visibility policies with attribute-based access control (ABAC) [24] without necessarily sacrificing simplicity of administration. This unification is important as companies may want to deny access to competitors or allow access to players outside the supply chain, such as auditors [21].

The enforcement of access control restrictions requires a preliminary authentication phase. The authentication phase is a protocol executed between the requestor and the data owner. It is assumed that players are uniquely identifiable and can reliably compute the predicate $\sigma(p_i, L(o_j))$. Moreover, a trusted third party (TTP) should be available for providing a PKI and RFID tag to players. Finally, it is also assumed that the communication between each player and the TTP is secure and authenticated and that RFID tags have re-writable permanent storage. The access control protocol operates in three steps: initialize, move, and authenticate [21].

Initialize: The trusted third party (T) sends to player p_i an RFID tag with identifier id which contains in its storage the signature $S_T(id, p_i)$.

Move: Player p_i wants to move an object to another player p_j. She appends to the storage on the RFID tag the recipient's identity (p_j) and her signature $Sp_i(id, p_j)$.

Authenticate: Let $s_1, ..., s_\kappa$ be the sequence of signatures stored on the RFID tag attached to object o_{id}. The requestor p_i sends as the verifiable predicate $\sigma(p_i, L(o_{id}))$ this sequence $s_1, ..., s_\kappa$. The verifier p_v verifies that

1) s_κ is equal to $Sp_{i_\kappa}(id, p_i)$.
2) $\forall\lambda \in \{2, \ldots, \kappa - 1\}$, s_λ is equal to $Sp_{i_\lambda}(id, p_{i_{\lambda+1}})$.
3) s_1 is equal to $S_T(id, p_{i_2})$.
4) $\forall\lambda \in \{1, \ldots, \kappa\}$, s_λ is valid signature from p_{i_λ}.

If all checks are successful, then p_v evaluates its policies to make the access decision.

Figure 4 provides an example of supply chain operations based on the above protocol. Assume a manufacturer M produces a good. She first starts the **Initialize** protocol by contacting the trusted third party T and requesting an RFID tag. T chooses a tag for the good with identifier g, stores on the tag memory $s_1 = S_T(g, M)$ and sends the RFID tag to M. M reads $\langle s_1 \rangle$ from the tag, stores it in its database and attaches the tag to the good g. Now M intends to ship g to its distributor D. She starts the **Move** protocol and appends $s_2 = S_M(g, D)$ to the tag's memory. M can then ship g to D. When D receives the good g, she reads $\langle s_1, s_2 \rangle$ from the tag and stores it in her database. Later D may ship g to the retailer R. In this **Move** protocol she appends $s_3 = S_D(g, R)$ to the tag's memory. R reads $\langle s_1, s_2, s_3 \rangle$ from the tag of the received good g and stores it in her database.

Now, assume that M wants to send a downstream request to R, (e.g., in order to collect sales data or analyze product returns). M and R run an **Authenticate** protocol. M reads $\langle s_1 \rangle$ from her database and sends it along with g, M to R. R's PEP verifies the data in $\langle s_1 \rangle$. It extracts M, g, and the number of signatures (1) from $\langle s_1 \rangle$. Then it looks up its corresponding predicate $\sigma(R, L(g)) = \langle s_1, s_2, s_3 \rangle$ in its database and similarly extracts R, g, and the number of signatures (3).

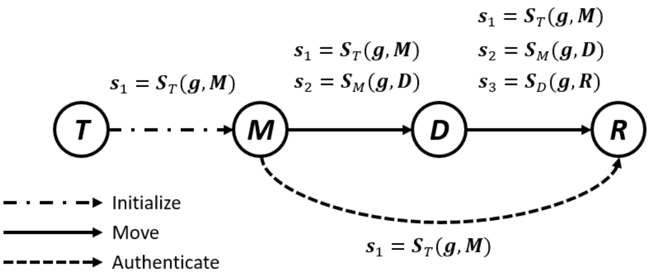


Fig. 4: An example of an object traversing a supply chain

IV. Supply Chain Access Control Enforcement

In this section, we illustrate how selective encryption (Section II) can be profitably used to enforce supply chain visibility policies (Section III). To properly enforce the visibility policy through selective encryption, each player p chooses her own private key k_p. Every time an object moves through the supply chain, each player who possesses it collects information about it. This information is represented as a tuple, which is then stored in a database in the cloud to simplify sharing with other players. Since the data collected about objects moving through supply chains are sensitive, tuples are encrypted before being inserted into the cloud database. To this aim, the player collecting the data chooses a unique symmetric encryption key k_o (and a corresponding public label l_o) for each object o that she handles, and uses such a key to encrypt the tuple t about o. Even if in principle each tuple t generated by player p could be encrypted with her own key k_p, we use a different key for tuple encryption to keep the role of player's private keys and that of encryption keys separate.

To illustrate how selective encryption can be used to enforce the visibility policy of supply chains, we analyze each of the steps of the access control protocol described in Section III. In the following discussion, we will refer our examples to the simple supply chain and object traversal illustrated in Figure 4. The manufacturer M has private key k_M, the distributor D has key k_D, and the retailer R has key k_R.

	t1	t2	t3
M	1	0	0
D	0	1	0
R	0	0	1

(a)

player	key
M	k_M
D	k_D
R	k_R

(b)

tuple	key
t_1	k_1
t_2	k_2
t_3	k_3

(c)

source	target	token
l_M	l_1	$k_1 \oplus h(k_M, l_1)$
l_D	l_2	$k_2 \oplus h(k_D, l_2)$
l_R	l_3	$k_3 \oplus h(k_R, l_3)$

(d)

Fig. 5: Access matrix and encryption policy subsequently to the move step

	t1	t2	t3
M	1	0	1
D	0	1	0
R	0	0	1

(a)

player	key
M	k_M
D	k_D
R	k_R

(b)

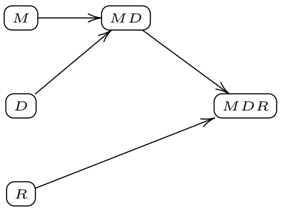
tuple	key
t_1	k_1
t_2	k_2
t_3	k_{MR}

(c)

source	target	token
l_M	l_1	$k_1 \oplus h(k_M, l_1)$
l_M	l_3	$k_3 \oplus h(k_M, l_3)$
l_D	l_2	$k_2 \oplus h(k_D, l_2)$
l_R	l_3	$k_3 \oplus h(k_R, l_3)$

(d)

Fig. 6: Access matrix and encryption policy after authentication of M for t_3



(a)

	t1	t2	t3
M	1	1	1
D	1	1	0
R	1	0	0

(b)

player	key
M	k_M
D	k_D
R	k_R

(c)

tuple	key
t_1	k_{MDR}
t_2	k_{MD}
t_3	k_M

(d)

source	target	token
l_M	l_{MD}	$k_{MD} \oplus h(k_M, l_{MD})$
l_D	l_{MD}	$k_{MD} \oplus h(k_D, l_{MD})$
l_{MD}	l_{MDR}	$k_{MDR} \oplus h(k_{MD}, l_{MDR})$
l_R	l_{MDR}	$k_{MDR} \oplus h(k_R, l_{MDR})$

(e)

Fig. 7: An example of key derivation graph, access matrix, and encryption policy

Initialize. At initialization time, a player p requests a set of RFID tags from the trusted third party and possibly starts to create objects. With reference to our example, after receiving the RFID tag from T, the manufacturer M creates an object g. She then generates tuple t_1, collecting information about the creation of g, and an encryption key k_1 to protect t_1. She then encrypts t_1 using key k_1 and stores it in the cloud. Since she is authorized to access tuple t_1, she also generates $token_{M,1}$ enabling her to derive k_1, and hence access t_1, from her own key k_M.

Move. When a player p_i sends an object to another player p_j, the receiver generates a tuple t to keep track of her operations over the object, and a new encryption key k to protect it. She then encrypts t using k and stores the encrypted tuple in the cloud. Since p_j can access t, she also generates a token enabling her to derive k from her own private key. With reference to our example, the manufacturer M sends g to D, who in turns forwards it to R. As a consequence of the first move action, D generates tuple t_2, encrypts it with a newly chosen key k_2, and computes token $token_{D,2}$. Similarly, as a consequence of the second move action, R generates tuple t_3, encrypts it with a newly chosen key k_3, and computes token $token_{R,3}$. The authorization and encryption policies resulting after the initialize and move steps of the protocol are illustrated in Figure 5. Note that, at this point, each tuple is visible only to its owner (i.e., the player who generated it).

Authenticate. Since each player p_i encrypted the information that she generated with a key that only she knows and can derive, any other player p_j who wants to access a tuple must first authenticate to the tuple owner to be granted access to the information. If player p_j successfully authenticates with p_i to access tuple t, then p_i will grant p_j access to t. Since each tuple is encrypted using a different key, to enforce such a policy change, p_i only needs to generate a token enabling p_j to derive the encryption key k used to protect t from her own private key. This guarantees that k can be derived by all the users in $acl(t)$, including p_j. Note that, it is not necessary to re-encrypt t, since simply adding the token enables p_j to derive k, while not disclosing other tuples to p_j. With reference to our example, assume that M would like to read t_3. To this aim, she needs to authenticate with R, who owns the tuple of interest, using the authentication protocol described in Section III. If the authentication is successful, R grants M access to t_3 and updates $acl(t_3)$=M to $acl(t_3)$=MR. Since k_3 has been used to protect only tuple t_3, it is sufficient to add token $token_{M,3}$ to the catalog to enable M to access t_3. Indeed, both M and R will be able to derive k_3, hence k_3 will become the key of the set $\{M,R\}$ of users. Since the computation of token $token_{M,3}$ requires knowledge of private information of both M (her private key k_M) and of R (k_3), the two players need to collaborate to compute it. To this aim, player M uses her private key k_M and the label of the key used to protect t_3, which is public, to compute $h(k_M, l_3)$. Then, she sends the result securely to R. Finally R computes the token as $k_3 \oplus h(k_M, l_3)$ and stores it in the public catalog. Figure 6 illustrates the authorization and encryption policies after the authentication of M for t_3.

Even if the approach illustrated above always permits to enforce the authentication step of the protocol in Section III, this may cause a growth in the number of nodes in the key derivation graph. Indeed, also tuples with the same *acl* would be encrypted with a different key. To limit the overhead due to key management, players can use the same encryption key for different tuples when they share the same *acl*, and dismiss keys when they are no longer used for tuple encryption. Granting or revoking access to tuples may then require to create new nodes or to delete old ones from the key derivation graph. As an example, assume that tuples $\{t_1, t_2, t_3\}$ belong to M (e.g., they represent three different objects that left M and eventually reached D and R), that D and R were granted access to t_1, and that D was granted access to t_2. Figure 7 illustrates the key derivation graph, the authorization policy, and the encryption policy for such a scenario. Assume now that, at a later time, R is granted access to t_2 and hence *acl*(t_2)=MDR. As there is already a node representing *acl*(t_2)=MDR in the key derivation graph, the tuple owner M just needs to decrypt t_2 with k_{MD} and re-encrypt it with k_{MDR}. After this policy update, node MD becomes obsolete since there is no tuple t with *acl*(t)=MD. Therefore, the node and all its incoming and outgoing edges can be removed from the graph (and hence the corresponding key and tokens are removed from the encryption policy). To guarantee correct key derivation, node MDR is directly connected with M, D and R. As illustrated above, players M and R will then need to cooperate to compute token *token*$_{R,MDR}$. However, as we already have a token *token*$_{R,MDR}$, we need to compute *token*$_{D,MDR}$ instead, which requires involvement of D. This is inconvenient for D, which already had access to t_2. To avoid the cooperation of players every time the token catalog needs to be updated as a consequence of an **Authenticate** step, the tuple owner p_j can agree in advance with each possible access requestor p_i an initial key k_i^j, which will be used by the requestor as a starting point in the derivation process. To guarantee that each player needs to manage one secret key only, we compute such a key as $k_i^j = h(k_i, l_j)$ where k_i is the unique private key of the requestor p_i. This way, M and D know $k_D^M = h(k_D, l_M)$, M and R know $k_R^M = h(k_R, l_M)$. Therefore, M can compute tokens *token*$_{D,MDR} = k_{MDR} \oplus k_D^M$ and *token*$_{R,MDR} = k_{MDR} \oplus k_R^M$ by herself.

V. Related Work

The problem of enabling data owners to specify and enforce access restrictions over data stored in the cloud, without the need for the data owner to filter access requests and to trust the cloud provider for authorization enforcement, has been recently widely studied. The solutions proposed to this problem follow two different strategies: attribute based encryption (e.g., [3], [14], [26]) and selective encryption (e.g., [2], [7]–[11]). Attribute-Based Encryption (ABE) [3], [14], [26] enforces attribute based access control policies over encrypted data. To this purpose, each tuple is encrypted with a key that only the users possessing attributes that satisfy the tuple policy can derive (or vice versa). Selective encryption [10], [11] instead translates the authorization policy into an equivalent encryption policy. Intuitively, each tuple is encrypted with a key known (or that can be derived) by all and only authorized users. The solution in [11] has been extended to enforce also write privileges [8], and to support the presence of multiple data owners [9]. The problem of efficiently manage policy update operations, limiting the data owner overhead, has been also studied [2], [11]. To this aim, the proposal in [11] delegates expensive re-encryption operations to the storing server. The approach in [2] instead specifically focuses on revoke operations, and introduces a solution that does not require complete re-encryption of revoked tuples.

A line of work related to our proposal is represented by access control enforcement using RFID. An extension of attribute based access control in RFID deployments has been previously formulated in [21]. However, in contrast to the work presented in this paper, the impact of the cloud and its requirement for efficient cryptographic capabilities, as we suggest, has not been analyzed. General approaches to RFID security and privacy are extensively surveyed in [17]. An alternative access control proposals for the RFID scenario at hand is represented by [4] which is considering the problem of specifying access only to part of the object data. Effectively this is expressible by the ABAC approach. The work in [16] formulates an access control approach in which the decision maker changes when an object is passed on from one party to another party, where we are uncertain about efficiency.

When providing confidentiality in the cloud by cryptographic means, querying of encrypted data is an important topic because encryption prevents the cloud provider from directly evaluating users queries due to no access to encryption keys. Furthermore, requiring the user to download the encrypted data and locally decrypt it before processing, nullifies the benefits of outsourcing the database to the cloud. Thus, we see the line of research on processing queries directly over encrypted data through specific encryption schemes, such as [5], [15], [22], [25], as an intriguing extension to the selective access approach formulated in this work.

VI. Conclusions

We have presented an approach based on selective encryption for enforcing downstream and upstream visibility policies in supply chain scenarios, where data about objects are stored at an external cloud service provider. The proposed approach enables companies to profitably use the sharing services offered by the cloud, while enforcing restrictions on the visibility of sensitive data. Our technique has the advantage of flexibility and enables the enforcement of arbitrary visibility policies over objects' data.

VII. Acknowledgment

This work was supported by the EC within the H2020 under grant agreement 644579 (ESCUDO-CLOUD).

References

[1] M. Atallah, M. Blanton, N. Fazio, and K. Frikken, "Dynamic and efficient key management for access hierarchies," *ACM TISSEC*, vol. 12, no. 3, pp. 18:1–18:43, January 2009.

[2] E. Bacis, S. De Capitani di Vimercati, S. Foresti, S. Paraboschi, M. Rosa, and P. Samarati, "Mix&Slice: Efficient access revocation in the cloud," in *Proc. of CCS*, Vienna, Austria, October 2016.

[3] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy Attribute-Based Encryption," in *Proc. of IEEE S&P*, Oakland, CA, May 2007.

[4] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," *Journal of Cryptology*, vol. 17, no. 4, pp. 297–319, Sep 2004.

[5] A. Ceselli, E. Damiani, S. De Capitani di Vimercati, S. Jajodia, S. Paraboschi, and P. Samarati, "Modeling and assessing inference exposure in encrypted databases," *ACM TISSEC*, vol. 8, no. 1, pp. 119–152, 2005.

[6] S.-Y. Chou and Y. Ekawati, "Cost reduction of public transportation systems with information visibility enabled by RFID technology," in *Proc. of CE*, Taipei, Taiwan, July 2009.

[7] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, and G. Livraga, "Enforcing subscription-based authorization policies in cloud scenarios," in *Proc. of the 26th Annual IFIP WG 11.3 Working Conference on Data and Applications Security and Privacy (DBSec 2012)*, Paris, France, July 2012.

[8] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, G. Livraga, S. Paraboschi, and P. Samarati, "Enforcing dynamic write privileges in data outsourcing," *Computers & Security*, vol. 39, pp. 47–63, November 2013.

[9] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, G. Pelosi, and P. Samarati, "Encryption-based policy enforcement for cloud storage," in *Proc. of SPCC*, Genova, Italy, June 2010.

[10] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, "Over-encryption: Management of access control evolution on outsourced data," in *Proc. of VLDB*, Vienna, Austria, September 2007.

[11] ——, "Encryption policies for regulating access to outsourced data," *ACM TODS*, vol. 35, no. 2, pp. 12:1–12:46, April 2010.

[12] B. L. Dos Santos and L. S. Smith, "RFID in the supply chain," *Communications of the ACM*, vol. 51, no. 10, pp. 127–131, October 2008.

[13] K. Finkenzeller, *RFID Handbook: Fundamentals and Applications in Contactless Smart Cards and Identification*. John Wiley & Sons, Inc., 2003.

[14] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proc. of CCS*, Alexandria, VA, October-November 2006.

[15] I. Hang, F. Kerschbaum, and E. Damiani, "ENKI: Access control for encrypted query processing," in *Proc. of SIGMOD*, Melbourne, Victoria, Australia, May – June 2015.

[16] A. Ilic, F. Michahelles, and E. Fleisch, "Dual ownership: Access management for shared item information in RFID-enabled supply chains," in *Proc. of IEEE PerCom Workshops*, White Plains, NY, USA, March 2007.

[17] A. Juels, "RFID security and privacy: A research survey," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 2, pp. 381–394, 2006.

[18] F. Kerschbaum, "Practical privacy-preserving benchmarking," in *Proc. of SEC*, Milan, Italy, September 2008.

[19] F. Kerschbaum, D. Dahlmeier, A. Schröpfer, and D. Biswas, "On the practical importance of communication complexity for secure multi-party computation protocols," in *Proc. of SAC*, Honolulu, HI, USA, March 2009.

[20] F. Kerschbaum, N. Oertel, and L. Weiss Ferreira Chaves, "Privacy-preserving computation of benchmarks on item-level data using RFID," in *Proc. of WiSec*, Hoboken, NJ, USA, March 2010.

[21] F. Kerschbaum, "An access control model for mobile physical objects," in *Proc. of SACMAT*, Pittsburgh, Pennsylvania, USA, June 2010.

[22] ——, "Frequency-hiding order-preserving encryption," in *Proc. of CCS*, Denver, Colorado, USA, October 2015.

[23] C. Kuerschner, F. Thiesse, and E. Fleisch, "An analysis of data-on-tag concepts in manufacturing," in *Proc. of MMS*, Munich, Germany, February 2008.

[24] NIST, "A survey of access control models," in *Privilege (Access) Management Workshop*, 2009.

[25] R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan, "CryptDB: Protecting confidentiality with encrypted query processing," in *Proc. of SOSP*, Cascais, Portugal, October 2011.

[26] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Proc. of EUROCRYPT*, Aarhus, Denmark, May 2005.

[27] S. Sarma, D. Brock, and D. W. Engels, "Radio frequency identification and the electronic product code," *IEEE Micro*, vol. 21, no. 6, pp. 50–54, November 2001.

[28] T. Staake, F. Thiesse, and E. Fleisch, "Extending the EPC network: the potential of RFID in anti-counterfeiting," in *Proc. of SAC*, Santa Fe, New Mexico, USA, March 2005.

[29] L. Weiss Ferreira Chaves and F. Kerschbaum, "Industrial privacy in RFID-based batch recalls," in *Proc. of 3M4EC*, Munich, Germany, September 2008.