Secure Cross-Domain Data Sharing Architecture for Crisis Management

Vaibhav Gowadia, Enrico Scalavino, Emil C. Lupu
Imperial College London
{vgowadia, escala, e.c.lupu}@imperial.ac.uk

Dmitry Starostin, Alexey Orlov
European Microsoft Innovation Center, Aachen
{dmitrys, alexeyo}@microsoft.com

Abstract

Crisis management requires rapid sharing of data among organizations responding to the crisis. Existing crisis management practices rely on ad hoc or centralized data sharing based on agreements written in natural language. The ambiguity of natural language specifications often leads to errors and can hinder data availability. Therefore, it is desirable to develop automatic data sharing systems. The need to share data during crises presents additional challenges, such as evaluation of security constraints in different administrative domains and in situations with intermittent network connectivity. We compare two different architectural approaches to develop secure data sharing solutions. The first approach assumes reliable network connectivity, while the second approach works in ad hoc networks. We then suggest a unified architecture that caters for both scenarios.

1 Introduction

Crisis management is the process of organizing response to incidents that seriously threaten people's lives or the environment. Examples of such incidents include road accidents, fires, etc. These incidents may evolve rapidly and need a quick and efficient response to limit the damages. Crisis management often requires access to sensitive data from many organizations or different administrative domains. For example, responders from different administrative domains may need to know the number of casualties, or the imminent dangers for the surrounding area, and medical records must be shared to provide care for the victims.

In the rest of this paper, we will indicate the organization where data originates as the *data provider* and the recipient organization as the *data consumer*. Sensitive data must be protected according to the security policies of the data provider

even after it has been disseminated to users in other organizations or different administrative domains. This usually requires mutual trust between the data provider and consumer. The data provider must ensure that its data will be protected after dissemination, while the data consumer needs to know what data it will be able to access. Organizations form *Data Sharing Agreements (DSA)* [26] to achieve these goals. A DSA is a signed contract stating each partner's obligations, which allows the partners to seek remedy (e.g. through legal means) for breaches of the contract. Establishment of a DSA allows the data provider to trust data consumers to enforce the agreed policies in their administrative domain.

In practice, DSAs are expressed in natural language (e.g., see [13]) and include the authorization policies for the shared data. Common practices of first-aid and public-safety agencies indicate that the evaluation and enforcement of DSAs are not automated and data is often shared informally [27]. Informal data sharing requires manual decision making. However, a manual process can cause delays and errors, which can in turn lead to deterioration of the crisis at hand. Effective and scalable crisis management requires an automated and efficient DSA evaluation and enforcement system.

Existing dissemination control architectures [18, 16, 2, 5, 10, 14, 15] are unable to address the requirements of cross-domain data sharing during crisis situations. First, they require recipients to contact the data provider or a pre-defined central policy evaluation authority to obtain access rights. However, responders may not be able to connect with the central authority while lending support in the crisis area. If, for example, the incident happens in a tunnel. The incident response command centers located in response vehicles may use long range communication equipment to exchange data with the outside world, while responders with mobile devices may have to depend on ad hoc links using short range communication of their devices. Ad hoc networks provide intermittent connectivity as the network nodes (responders) are always in motion. A remote policy evaluation authority could be unreachable from the responder's location if the network is temporarily partitioned. This limitation can be fatal in crisis management scenarios. It is therefore necessary to cater for situations when data may be shared through an ad hoc network among rescuers, or manually using portable media such as data sticks. To effectively respond to crises, data must be protected in ways that do not require communication with entities outside the responder's reach.

Another limitation of the existing dissemination control architectures is that policies are unilaterally defined by the data provider who may have poor or no knowledge of user credentials and contextual information available in the data consumer's administrative domain. This severely restricts the expressiveness of the policies that can be enforced in practice.

Example 1.1 Consider a crisis scenario where a leakage of a toxic chemical is threatening a neighborhood and the Police coordinates the evacuation of people living nearby. Responders collect sensitive personal and medical information of the evacuees, and need to share it with employees of the Local Government to coordinate medical support for the evacuees. It is responsibility of the Police to protect the collected sensitive data. Following the *need to know* principle, the Police should give access to only those members of the Government staff that are assigned to the incident at hand. To specify such an access policy the data provider (Police) must know how to identify personnel assigned to the incident in the Local Government's administrative domain. The data consumer (Local Government) must understand the conditions required by the data provider's policy and be able to provide matching credentials. In other words, the data sharing partners must agree on a common/shared vocabulary as part of their DSA. □

This paper presents a data sharing architecture based on the concept of DSAs that caters for data access with intermittent connectivity. We identify a policy evaluation and key distribution scheme suitable for ad hoc / opportunistic networks [19, 28, 9] that allows users to obtain permissions without communicating with a central policy evaluation authority. We discuss how the architecture enables *usage control* [17]; i.e., continuous control over data access.

The remainder of this paper is organized as follows: Section 2 discusses related works and their limitations. In Section 3 we introduce the concepts of DSA and usage control policy. Section 4 discusses the design options for a cross-domain data sharing architecture. We also propose a unified architecture that addresses the requirements of data sharing in crisis scenarios. Section 5 describes the deployed components and their behavior. We give an overview of our implementation in Section 6. In Section 7, we discuss further design options and features. Finally, we conclude in Section 8 where we also discuss how the proposed architecture addresses the stated requirements.

2 Related Work

Dissemination control systems are also known as Digital Rights Management (DRM) and Enterprise Rights Management (ERM) systems [5, 15, 25] and share many common design principles and functionalities. The sensitive data is cryptographically protected and associated with access policies. These policies are evaluated by a central trusted authority (TA), who is often the originator itself. The TA evaluates recipients rights and issues the decryption keys to them (see Figure 1). This kind of architecture suffers from limitations that make its use inappropriate in crisis management scenarios: 1) The TA is statically defined and any change requires

manual intervention; 2) if the TA is not available, the users cannot access the data; 3) cross-domain evaluation is hard to achieve due to the limited knowledge of the data consumer's domain.

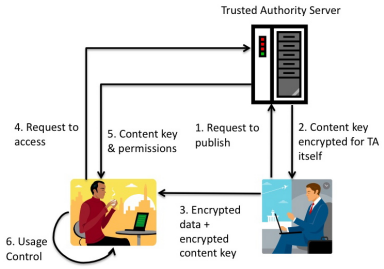

Figure 1: Deployment of Centralized Rights Distribution Architectures

Park et al. [18] presented an analysis of possible dissemination control architectures based on three elements: virtual machine (enforcement layer), control set (list of access rights and usage rules), and distribution scheme (push or pull). They also identified three types of control sets: fixed, embedded, and external. A fixed control set is hard coded into the virtual machine. An embedded control set is sent with the protected data and an external control set resides at a remote location. However, they do not explain how rights can be determined during cross-domain evaluations.

Adam et al. [1] proposed a data sharing architecture that uses an intermediate coordinator service in each organization. Data can be shared with other organizations in push or pull mode. In push mode, data can be shared at inter-agency level between the coordinator services, and at intra-agency level between a coordinator service and recipients within its organization. Whereas in pull mode data is shared only at inter-agency level. In their architecture, after the coordinators have exchanged data the originator does not retain any control over it. Moreover, the proposed solution is unusable when connectivity is absent.

Since it is not feasible to use a central policy evaluation authority in crisis management, it is necessary to explore other rights distribution architectures that do not require communication with a central server.

Access Hierarchy based Encryption (AHE) [4] defines a hierarchy of *access-levels*. Each access-level is associated with a secret key and a public label. while each edge is assigned a public value. The secret key of a node in the hierarchy can be derived in an offline mode if the secret key of its parent node is known.

Attribute Based Encryption (ABE) [20] can be considered as an evolution of Identity Based Encryption (IBE) [7], where documents are encrypted using an at-

tribute based policy and a TA's public key. The TA generates a secret key for each user based on the user's attributes so that they are able to decrypt data only if their attributes satisfy the attribute based policy used for the encryption. Again, attribute-based keys must be disseminated before any access request.

The Policy-based Authority Evaluation Scheme (PAES) [22] allows a data originator to specify a hierarchy of authorities trusted to correctly evaluate its policies. The set of authorities trusted to evaluate the usage control policies is not statically defined, but designated by a trust policy evaluated by a higher level authority. The set of higher level authorities is recursively defined by policies. The recursion terminates by a set of statically defined authorities at the root-level of the hierarchy. As a result this protocol allows recipients to choose a trusted authority in a more flexible manner.

3 Definitions

The core ingredients of a DSA are 1) the scope of the agreement, 2) the attribute vocabulary, 3) the security policies, and 4) the penalties for violating the agreement.

Definition 3.1 (Data Sharing Agreement)
A Data Sharing Agreement *dsa* is a 6-tuple (id, A, S, T, P, V) where *id* is a unique identifier, A is an attribute vocabulary describing subject, data, and context attributes, S is the scope of the agreement, T is a set of authorities trusted for providing user and context attributes, P is a set of usage policies that must be enforced, and V is a set of violation procedures applicable whenever P is violated. The scope of the agreement is defined as a 4-tuple (E, O, t_s, t_e) where E is the set of entities signing the DSA, O is a sets of conditions over data attributes that must be satisfied by all data items to which this agreement is applicable, t_s and t_e are the dates and times at which the agreement becomes effective and expires respectively.

When an entity agrees to a DSA it promises to enforce policies equivalent to P for the data provided by the partner entities. This ensures that the security requirements of the data provider are satisfied.

DSAs may include additional specifications, such as procedures to resolve disputes, to revise the agreement etc. Discussion of these components and violation procedures are outside the scope of this paper.

Park et al. [17] described a formal model for usage control based on authorizations, obligations, and conditions. Conditions describe constraints over subject, data, and contextual attributes. To provide continuity of control the usage control policy must be reevaluated if any of the attributes change. This requires monitoring

of the attributes used in access conditions. However, monitoring all attributes for a large set of documents can consume significant computing and energy resources. We therefore observe that attributes may be either long-lived (*persistent*) or short-lived (*volatile*). For example, a user's id or group may be considered long-lived, whereas a user's location is short-lived as it may change frequently during an access session. Thus, in our definition of usage control policy, we allow the policy author to clearly separate the long and short lived access conditions so that only the latter are monitored.

Definition 3.2 (Usage Control Policy)
A usage control policy p is a 5-tuple (s, o, a, c, c_m) representing a permission given to subject s to perform action a on object o, when the conditions c and c_m are satisfied. To allow access c must be satisfied at the moment of the access request, whereas condition c_m must hold for the entire duration of the usage. This means c_m must be monitored after access is granted and the usage must be interrupted as soon as c_m becomes false.

Although the DSA partners agree on a common vocabulary, the attributes used in the DSA may still not match with the credentials issued by the organizations. Thus, the architecture should either translate the usage policies of each DSA to a domain-specific enforceable version, or translate the credentials for each access request depending on the applicable DSA. We take the first approach because the translation needs to be done only once for each DSA.

Example 3.1 Let us consider a DSA between the Police and a Local Government that specifies that access to personal data of victims in an incident should be restricted to responders assigned to that same incident. Moreover, access must be allowed only *while* the responder is located in a danger area designated as *level1*. Figure 2 gives an example of the DSA translated into an enforceable policy for the Police administrative domain.

The first two statements of the example policy declare the persistent and volatile attributes and specify their data types. Authority P is declared by specifying the identifier of the service that issues credentials on its behalf. The credential types used in the policy are declared by specifying their names and the lists of attributes they can contain. Credentials are then defined as instances of credential types issued by a trusted authority. User roles are declared by specifying a condition over the credentials users may possess. Finally, a usage control policy specifies that a subject with role "responder" can read a data item of category "personal". The monitored condition c_m is expressed in the *while* clause of the policy, and other access conditions are specified in the *when* clause. □

```
volatile attribute locCategory string;
persistent attribute uid, group, incidentId string;
authority P = "police";
credtype Authentication(uid, group);
credtype Location(category);
credtype MissionAssignment(uid, incidentId);
credential AuthnToken = Authentication signedby P;
credential LocationToken = Location signedby P;
credential MissionToken = MissionAssignment signedby P;
role responder requires AuthnToken.group=="constable";
authorization a1 = allow read()  target ( dataCategory=="personal" )  to responder
   when ( MissionToken.incidentId == object.incidentId and MissionToken.uid==AuthnToken.uid )
   while ( LocationToken.locCategory=="level1" );
```

Figure 2: Example of an Enforceable Policy

4 Architecture

One of the main threats for disseminated data is the possibility for an attacker to bypass a genuine policy enforcement point (PEP) and access the data directly. To mitigate such threat, the sensitive data must always be kept encrypted when stored or transmitted. Only a genuine PEP should be able to obtain correct decryption keys when permitted by a trusted policy decision point (PDP). We categorize DRM/ERM architectures based on whether users must interact with other entities to receive decryption keys *after* or *before* receiving protected data. We first discuss and compare these two approaches, and identify their advantages and limitations. We then present a unified architecture combining the benefits of both approaches.

4.1 Interactive Data Sharing Architecture (IDSA)

In the IDSA architecture recipients interacts with their organization's server to obtain access *after* receiving the protected data. If the data provider wants to evaluate access requests of recipients in other organizations, then additional infrastructure is needed to map the recipients' credentials to the credentials it understands. We observe that a DSA between the data provider and data consumer establishes trust which allows the data consumer organization to evaluate access requests of its users. Thus, mapping of credentials is not needed and architectures presented in this paper take the approach of translating the DSA into policies enforceable by the DSA partners.

IDSA is an extension of traditional DRM/ERM model and is described first to illustrate how cross-domain operation can be achieved. It does not however cater for the need to access protected data when connectivity is intermittent.

To evaluate the usage policies, a policy engine needs to identify the data for

which access has been requested. Content-based rules cannot be applied because the PEP is not yet able to decrypt the protected content. To overcome this problem, it is necessary to package metadata with the protected document. Also, the metadata must be protected against unauthorized modifications. We here assume the metadata is specified as a list of attribute and value pairs. For example, metadata for personal data sharing during a crisis incident may be represented as [(incidentId,"123"),(dataCategory,"personal")].

Let k denote a symmetric encryption key and $\{D\}_k$ the data D encrypted with k. We also refer to k as the *content key*. PK_e and PK_e^{-1} denote the public and private keys of an entity e. $Cert_e$ denotes the public key certificate of e. Let the user Bob working for organization B request dissemination of a data item under a DSA DSA_{AB} between organizations A and B. The data is described by the metadata M. A protocol (see Figure 3(a)) for securely sharing D with a user Charles who works for organization B can then be described as follows:

Bob requests to protect data for recipients in organization B:

1. $Bob \rightarrow Server_A : \{k, DSA_{AB}, M\}_{PK_A}, Credentials_{Bob}$
2. $Server_A \rightarrow Bob : \{k, DSA_{AB}, M\}_{PK_B}$
3. $Bob \rightarrow Charles : \{D\}_k, \{k, DSA_{AB}, M\}_{PK_B}$

Charles requests access to the protected data:

1. $Charles \rightarrow Server_B : \{k, DSA_{AB}, M\}_{PK_B}, Credentials_{Charles}, Cert_{CharlesPEP}$
2. $Server_B \rightarrow Charles : \{k, p, M\}_{PK_{CharlesPEP}}$

We assume that all messages are signed by their senders. The content key is encrypted for $CharlesPEP$ (PEP of the device used by Charles). Thus, the content key is not directly accessible to the end-user. The PEP uses the content key to allow access to the user when authorized according to the usage control policy p.

IDSA is an extension of ERM systems such as the Microsoft Rights Management Services [15] for cross-domain operation. Moreover, IDSA allows specification of more expressive policies. The key differences in the extension include the use of metadata for policy-evaluation, the presence of a DSA, and the distribution of policy evaluation authorities based on trust between the organizations.

4.2 Non-Interactive Data Sharing Architecture (NIDSA)

In the NIDSA architecture users obtain decryption keys *before* receiving protected data. Thus, users are not required to interact with any entity after receiving data. Also, the same key allows the users to access any document for which they have permissions.

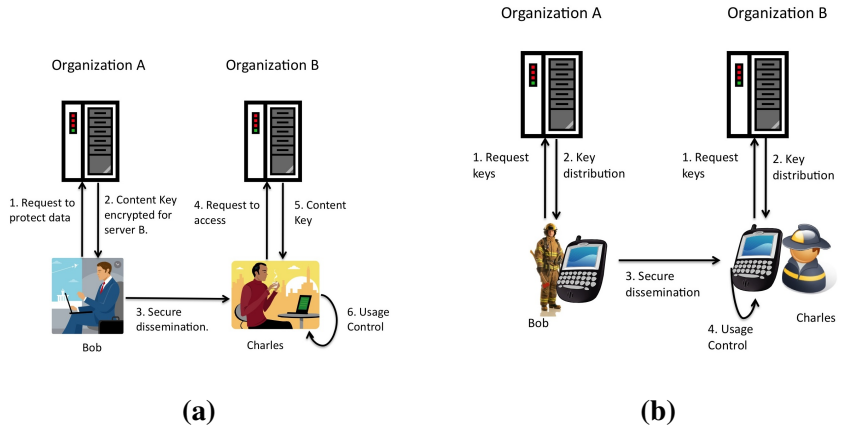

Figure 3: Deployment of (a) Interactive and (b) Non-Interactive Secure Cross-Domain Data Sharing Architectures.

To encrypt data for users in another organization, it must be possible to use a public key k_g such that members of a group g will already have the corresponding private key k_g^{-1}. It is desirable that users obtain the group's private keys they are authorized to possess before data is received. In addition, each user u has its own public-private key pair PK_u, PK_u^{-1}. In this architecture, all users must know the public keys of all groups in order to encrypt data for other users.

Let $K_{g(Bob)}^{-1}$ and $K_{g(Charles)}^{-1}$ be the set of private group keys that Bob and Charles possess. A protocol for securely sharing data D using NIDSA is shown in Figure 3(b). It may be noted that in NIDSA the recipient obtains access keys in step 2 before dissemination step while in IDSA the recipient obtains the access keys in step 5 after dissemination. The protocol can be described as follows:

1. $Bob \rightarrow Server_A : Credentials_{Bob}$
   $Charles \rightarrow Server_B : Credential_{Charles}$
2. $Server_A \rightarrow Bob : \{K_{g(Bob)}^{-1}, p\}_{PK_{BobPEP}}$
   $Server_B \rightarrow Charles : \{K_{g(Charles)}^{-1}, p\}_{PK_{CharlesPEP}}$
3. $Bob \rightarrow Charles : \{D\}_k, \{k, p\}_{k_g}, \{id_{dsa}, M\}_{Signed}$

Charles is authorized to access data if $k_g \in K_{g(Charles)}^{-1}$. Moreover, Charles does not need to contact a server to access data. The usage control policy p still needs to be enforced locally. For example, it is often desirable to restrict access using additional context, e.g., responders may view the data of only those incidents they are assigned to.

It is possible to build a NIDSA scheme based on *Cipher-text Attribute Based Encryption* (CPABE) [6]. Possession of an attribute in ABE is comparable to having a group private key in NIDSA. However, existing ABE schemes are not an ideal solution for crisis management because they are computationally very intensive for mobile devices that responders need to use.

The NIDSA architecture can also be used with symmetric key encryption schemes such as AHE [4] by considering $k_g = k_g^{-1}$, and each access-level in AHE as a group in NIDSA. To protect data using AHE, the data provider must encrypt data using a key that the authorized recipients can derive. An implementation of NIDSA architecture using AHE will require generation of an access hierarchy which includes members of multiple administrative domains.

Note that the NIDSA architecture does not cater for access revocation, while in an IDSA revocation checks can be done at the organization's servers as the recipients must request permissions for each protected item. In the NIDSA architecture, revocation can be realized only by forcing a key renewal at regular intervals and by not providing the fresh set of decryption keys to the revoked users. However, during the interval of validity of a set of keys the user may still decrypt any new document she receives. This is a trade-off that needs to be made to gain the ability of accessing data in offline mode. While NIDSA caters for crisis management requirements, it is still desirable to use IDSA during normal operation. Therefore, an architecture providing a comprehensive secure data sharing solution must unify NIDSA and IDSA.

4.3 Unified cross-domain Data Sharing Architecture (UDSA)

To benefit from the diverse characteristics of IDSA and NIDSA, we build UDSA by decentralizing IDSA and using some features of NIDSA. Similar to NIDSA, users in UDSA are assigned to different groups and each group has a public-key and a private-key. We here assume that each responder has a wireless device capable of short-range wireless communication (e.g., a PDA or a smart phone). We also assume that the persistent user credentials and the public keys of all organizations that are part of DSAs are already stored on the user's device.

Unlike NIDSA, the key pairs for groups are generated on a per-incident basis in UDSA. We assume one of the incident response vehicles acts as an initial source for creating and distributing keys to be used for that specific incident. The key distribution source generates a public-private key pair for each group used in the organization's DSAs. The responders exchange data, keys, and certificates when they come into communication range of each other. This communication together with their mobility in the incident area, allows the responders to cope with intermittent network connectivity.

The ad hoc distribution of rights in form of private-keys of groups is controlled by policies, inspired by the Policy-based Authority Evaluation Scheme (PAES) [22]. The private keys are always stored by the PEP securely and are not directly accessible to end-users (discussed further in Section 7). The PEP uses a group's private key on behalf of a responder only if 1) she is a member of that group, or 2) she is trusted to evaluate membership of other users to that group, according to a PAES policy-hierarchy.

PAES allows definition of a hierarchy of authorities, where each authority is trusted to evaluate and distribute rights. PAES can be implemented for crisis management by allowing responders to act as authorities. The implementation of PAES for crisis management [23] can be adapted to work during normal operation by including the organizational control center as an authority. A deployment of UDSA in a crisis area is illustrated in Figure 4.

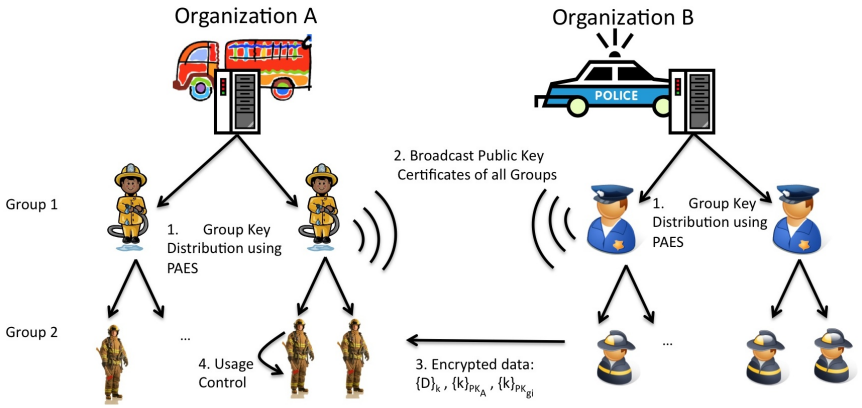

Figure 4: A Deployment of Unified Cross-Domain Data Sharing Architecture

5 UDSA Components

The goal of having a unified architecture is to provide secure access to data in diverse scenarios. In our architecture, the PEP comprises the application and a Data Protection Object API (DPOAPI). The DPOAPI provides a common set of functionalities including cryptographic functions and interactions with the PDP. The application is responsible for intercepting access requests and shares the responsibility to enforce access decisions with the DPOAPI.

Usage control over disseminated data requires distributed policy evaluation and enforcement components. First, a user must contact a control center to obtain a *use-*

license. In UDSA, a control center can be the user's organization server or a trusted peer according to the PAES policy hierarchy. If the user obtains the use-license using adhoc mode, the license contains a group key. However, if the user obtains the use-license for a specific protected data item then the use-license contains the content key. In addition, a use-license contains policies to be evaluated locally and an expiration date after which a new use-license must be obtained. After a user obtains a use-license, the usage control policies in it must be evaluated to check the user's authorization for each access session and monitor the access if required. Note that the use-licenses are protected for the PEP on the user machine. We now discuss these two steps in further detail.

5.1 Obtaining A Use-License

The difference between evaluating a use-license request in the IDSA and NIDSA is that for the latter the process depends only on the user-attributes while in the former it also depends on the metadata of the protected data. Having recognized this difference it is possible to design policy and enforcement components such that they can seamlessly and automatically switch to ad hoc mode when network connectivity is not available. Existing policy engines can be easily modified to operate in ad hoc mode by distinguishing between conditions that depend on metadata and conditions that do not.

Figure 5 shows the interactions for obtaining a use-license. The optional parameters are marked with (*) in the figure. Before requesting a use-license, the PEP needs to collect the user credentials necessary for the evaluation. Collection of all possible credentials can be time consuming. Thus, the PDP in USDA can be queried for the credentials needed to evaluate a user's request. The PDP returns a list of credential requirements, i.e. of pairs (credential type, issuer). A credential issuer is the authority trusted to specify values for a specific credential type. For example, for the policy shown in Figure 2 the credential requirements are [(Authentication, "police"), (Location, "police"), (MissionAssignment, "police")]. The PEP then asks the Policy Information Point (PIP) to obtain the required credentials. The PIP is configured with transport-level knowledge (e.g. URLs of credential services) and certificates of trusted entities, so that it can securely obtain credentials. It can also cache credentials for use in offline mode. The duration for which a cached attribute value may be used depends on whether it is persistent or volatile. The PDP of peer/server then evaluates specified DSA and user credentials to determine whether a use-license can be given to the user.

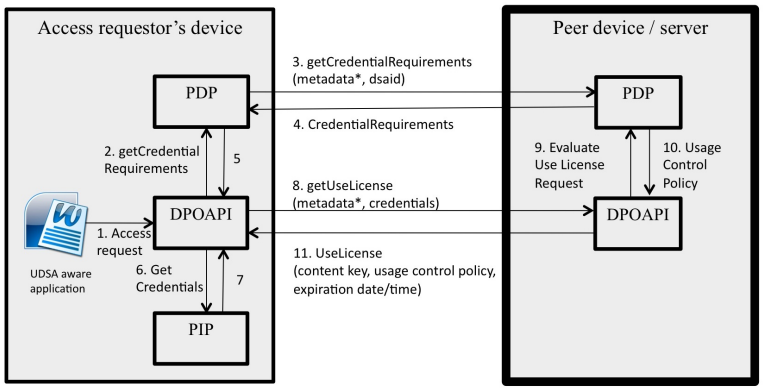

Figure 5: Interactions to obtain a Use License

5.2 Usage Control

Figure 6 illustrates the interactions to evaluate access when a user has a valid use-license. Implementation of usage control requires that the PDP, DPOAPI and the controlled application implement listener interfaces to handle changes in the monitored attributes. On access evaluation, if the policy granting access to the user requires monitoring, the PDP subscribes to the PIP to receive notification about changes in the monitored attributes. The PIP then subscribes to local or remote attribute authorities. Note that in ad hoc mode we require remote authorities for monitored attributes to be within the local network and communication with them should be possible using opportunistic routing. When a monitored attribute changes, the PIP sends an *attribute changed event* to the PDP. The PDP is session-aware and keeps track of sessions that need to be re-evaluated when subscribed attributes change. Thus, the PDP determines the sessions that must be reevaluated and asks the DPOAPI to send the access evaluation requests for those sessions. The DPOAPI obtains the cached copy of the user credentials from the PIP and sends the evaluation requests to the PDP. If the permissions returned by the PDP are different from the previous permissions, the DPOAPI sends a *permission changed event* to the application. The application is trusted to terminate the ongoing accesses, while the DPOAPI will stop providing decrypting data for the application if the user has lost all permissions.

The monitored conditions may need to be reevaluated several times during an access session. It is efficient to evaluate the monitored conditions because communication between the PEP and PDP are local and no network communication is needed after an attribute changed event is received.

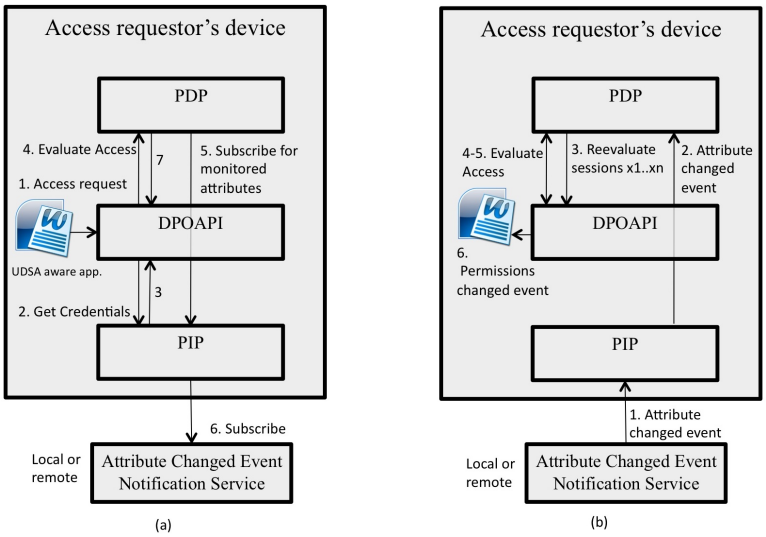

Figure 6: Usage control (a) evaluation for new session and (b) revaluation in an ongoing session.

6 Implementation

We have implemented the UDSA architecture and tested with development of two UDSA-aware applications. The PDP components of the architecture were developed at Imperial College London, and the PEP and PIP components were developed at European Microsoft Innovation Center (EMIC). The PDP component for end-user devices has been implemented as a Java library, and the server-side PDP component is exposed as a Java web service. The DPOAPI and PIP components are implemented using C#. Interoperability between Java and C# components is achieved using JNBridge [12].

The two UDSA-aware applications demonstrate use of the architecture in different scenarios. The first application provides secure access to XML documents containing information about an incident. The data-protection mechanism implemented is flexible and allows us to restrict the access to selected parts of the document. In addition, the access to the document can be restricted based on the user's location, role, and status of the incident. User's location is determined by triangulation of wifi tags (Aeroscout tags [3]) carried by the user. The user's identity and location credentials are issued by a security token service (STS) implemented using the Geneva Framework [8]. The second application instead provides secure access to sensitive scientific data in a typical enterprise scenario.

7 Discussion

A pragmatic data sharing architecture for crisis management should allow a responder to override the security policies to access data in high risk situations. Implementation of such *break-the-glass* security mechanism should be accompanied with the creation of an activity log that can be reviewed later.

In general, when users have complete privileges on their devices, it is necessary to rely on the *trusted computing* (TC) technology [24, 11] to ensure secrecy of the encryption keys (e.g. PEP's private key) and the integrity of the policy and enforcement infrastructure. TC provides hardware-based security and allows only trusted software to access *sealed* data such as keys. Sandhu et. al [21] describe several architectures based on TC to provide policy enforcement and implementation. However, in case of Government agencies responding to a crisis situation or large corporations the user privileges are often restricted and well managed. In such cases, it is not necessary to rely on TC. The safety of keys and integrity of the policy and enforcement infrastructure can be provided by restricting user access over them.

Our architecture also allows the end users to freely redistribute protected data to others. However, to access the protected data each recipient needs to obtain a use-license from one of the designated policy evaluating authorities after presenting her credentials.

UDSA uses an access revocation similar to IDSA. On use license expiration the PEP requires the user to renew the use-license to continue the access. This allows the trusted policy evaluation authorities to check for revocations.

8 Conclusions

In this paper we presented an architecture for secure data sharing in crisis scenarios. The architecture uses a flexible policy evaluation scheme that provides automated evaluation of data sharing requests. Automation allows the architecture to provide faster and secure data sharing than manual data sharing practices. In addition, usage control on disseminated data can be enforced without need to connect with organizational infrastructures because users can be evaluated within their local network. This allows the architecture to cater for data sharing among users with intermittent connectivity.

We have also shown that it is possible to use the same architecture for scenarios with and without reliable network connectivity. The UDSA architecture achieves this goal by using distributed policy evaluation and establishing the required trust relationships using the concept of DSAs.

References

[1] N. Adam, A. Kozanoglu, A. Paliwal, and B. Shafiq. Secure information sharing in a virtual multi-agency team environment. *Electron. Notes Theor. Comput. Sci.*, 179:97–109, 2007.

[2] Adobe. A primer on electronic document security, 2004. White Paper. http://www.adobe.com/security/pdfs/acrobat_live-cycle_security_wp.pdf Accessed 22-Sep-2009.

[3] Aeroscout. http://www.aeroscout.com/content/tags.

[4] M. J. Atallah, K. B. Frikken, and M. Blanton. Dynamic and efficient key management for access hierarchies. In *CCS '05: Proceedings of the 12th ACM conference on Computer and communications security*, pages 190–202, New York, NY, USA, 2005. ACM.

[5] Authentica. Enterprise rights management for document protection, 2005. White Paper.

[6] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. In *SP '07: Proceedings of the 2007 IEEE Symposium on Security and Privacy*, pages 321–334, Washington, DC, USA, 2007. IEEE Computer Society.

[7] D. Boneh and M. Franklin. Identity based encryption from the Weil pairing. In *Proc. 21st Annual Int. Cryptology Conference*, pages 213–229, Santa Barbara, USA, 2001.

[8] M. L. Bustamante. A better approach for building claims-based wcf services. MSDN magazine, December 2008. http://msdn.microsoft.com/en-us/magazine/dd278426.aspx.

[9] M. Conti and S. Giordano. Multihop ad hoc networking: The reality. *IEEE Communications Magazine*, 45(4):88–95, April 2007.

[10] EMC. Emc documentum information rights management: Overview of technical architecture, 2006. White Paper.

[11] E. W. Felten. Understanding trusted computing: Will its benefits outweigh its drawbacks? *IEEE Security and Privacy*, 1(3):60–62, 2003.

[12] JNBridgePro. http://www.jnbridge.com/jnbpro.htm.

[13] Leeds Primary Care Trust. Appendix 3 - Example of an Information Sharing Agreement. In Leeds Interagency Protocol for Sharing Information., 2008. http://www.leedspct.nhs.uk/?pagepath=About Us/Information Sharing/Protocol. Accessed 10-Dec-2009.

[14] Liquid Machines. Liquid Machines and Microsoft Windows Rights Management Services (RMS): End-to-end Rights Management for the Enterprise, 2006. White Paper.

[15] Microsoft. Technical overview of windows rights management services for windows server 2003, 2005. White Paper. URL: download.microsoft.com/download/8/d/9/8d9dbf4a-3b0d-4ea1-905b-92c57086910b/RMSTechOver-view.doc.

[16] M.-C. Mont, S. Pearson, and P. Bramhall. Towards accountable management of identity and privacy: Sticky policies and enforceable tracing services. In *DEXA Workshops*, pages 377–382, 2003.

[17] J. Park and R. S. Sandhu. The $UCON_{ABC}$ usage control model. *ACM Trans. Inf. Syst. Secur.*, 7(1):128–174, 2004.

[18] J. Park, R. S. Sandhu, and J. Schifalacqua. Security architectures for controlled digital information dissemination. In *16th An. Computer Security Applications Conf. (ACSAC)*, pages 224–, New Orleans, USA, 2000. IEEE Computer Society.

[19] R. Ramanathan, R. Hansen, P. Basu, R. Rosales-Hain, and R. Krishnan. Prioritized epidemic routing for opportunistic networks. In *Proc. of the 1st int. MobiSys Workshop on Mobile Opportunistic Networking (MobiOpp)*, pages 62–66, New York, USA, March 2007. ACM.

[20] A. Sahai and B. Waters. Fuzzy identity-based encryption. In *24th Int. Conf. on the Theory and Applications of Cryptographic (EUROCRYPT)*, volume 3494 of *LNCS*, pages 457–473. Springer, 2005.

[21] R. S. Sandhu, K. Ranganathan, and X. Zhang. Secure information sharing enabled by Trusted Computing and PEI models. In *ASIA CCS*, pages 2–12, 2006.

[22] E. Scalavino, V. Gowadia, and E. C. Lupu. PAES: Policy-based Authority Evaluation Scheme. In *Proceedings of the 23rd Annual IFIP WG 11.3 Working Conference on Data and Applications Security XXIII*, pages 268–282, Berlin, Heidelberg, 2009. Springer-Verlag.

[23] E. Scalavino, G. Russello, R. Ball, V. Gowadia, and E. C. Lupu. An Opportunistic Authority Evaluation Scheme for Data Security in Crisis Management Scenarios. In *Proceedings of Asia CCS*, 2010.

[24] S. Schoen. Trusted Computing: Promise and Risk, 2003. URL: http://www.eff.org/files/20031001_tc.pdf.

[25] A. Secure. Choosing an enterprise rights management system: Architectural approach, 2007. URL: www.windowsecurity.com/uplarticle/Authentication_and_Access_Control/ERM-architectural-approaches.pdf.

[26] V. Swarup, L. Seligman, and A. Rosenthal. A data sharing agreement framework. In *ICISS*, pages 22–36, 2006.

[27] United Kingdom. Cabinet Office. Emergency preparedness - guidance on part 1 of the civil contingencies act 2004, its associated regulations and non-statutory arrangements. URL: http://www.cabinetoffice.gov.uk/media/131903/emergprepfinal.pdf.

[28] A. Vahdat and D. Becker. Epidemic routing for partially connected ad hoc networks. Technical Report CS-2000-06, Duke University, April 2000.