Managed Information Objects for Secure Data Sharing

Abstract

Common approaches for protecting disseminated data (such as Digital Rights Management solutions) against privacy violations use a client application at the host receiving the disseminated data to enforce the privacy policies associated with the data. An alternative approach for protecting disseminated data is to bundle together the data, metadata (policies), and a mechanism that enforces the policies included in the metadata, into a Managed information object (MIO) or active bundle (AB). The common approaches trust the client application to enforce the privacy policies associated to the data. The MIO/AB approach trusts the client host to execute the mechanism that enforces the associated policies. The proposal discusses the MIO/AB approach and shows that bundling data with metadata and a policy enforcement mechanism provides enhanced security and reduces the risk of privacy violations for disseminated data.

1. Introduction

Today, we are living in a digital economy based on the exchange of information. There is petabytes of information created, processed, exchanged, consumed and shared everyday around the world between organizations and people. IT has been progressing rapidly and with the emergence of models such as cloud computing and service oriented architecture, it has become even easier to access and share information. But the models that facilitate this information sharing and exchange can also make it easier for unauthorized and malicious parties to get access to personal and sensitive information. This raises important security and privacy concerns. Failure to disseminate and share data properly can lead to data leakage and unauthorized access resulting in data loss and privacy violation. The problem of data leakage and exposure has been discussed in detail in [1].

Motivation

More and more common everyday objects are becoming smart and acquiring cyber presence by the use of sensors and embedded devices. Seamless interconnection of such smart things is a step towards making Pervasive Computing a reality. This model can be used to support wide range of Ubiquitous Computing applications such as smart home, pervasive healthcare monitoring, smart grid etc. We consider smart home for discussion and motivation.

Smart Home is a regular home equipped with hi-tech sensors and technologies that integrate with information-based services to automate various home activities, provide seamless services and ease of access to the information that enhances occupants' quality of life. These sensors and technologies integrate together to form different components of smart home such as smart door, smart bed, smart mailbox, smart closet, smart refrigerator, smart pantry, smart floor, smart bathroom, etc [2]. For instance, things like a smart bed can monitor a person's health, her sleep patterns and keep track of sleepless nights and automatically send this information to healthcare providers. Thinks like smart refrigerator, smart pantry, smart drawers, smart closets can monitor content availability and consumption, detect expired items (food, medicines), provide counseling and services like automated replenishing of goods. Similarly a smart bathroom can regulate water temperature, measure occupant biometrics such as body weight and temperature and notify the service center for automated refills of toiletries. A smart floor can identify and track the location of all house occupants, detect and report emergency like an occupant fall. A smart security system can continuously monitor windows, doors and track potential emergencies, query the resident if it suspects a problem, and report to emergency services for help.

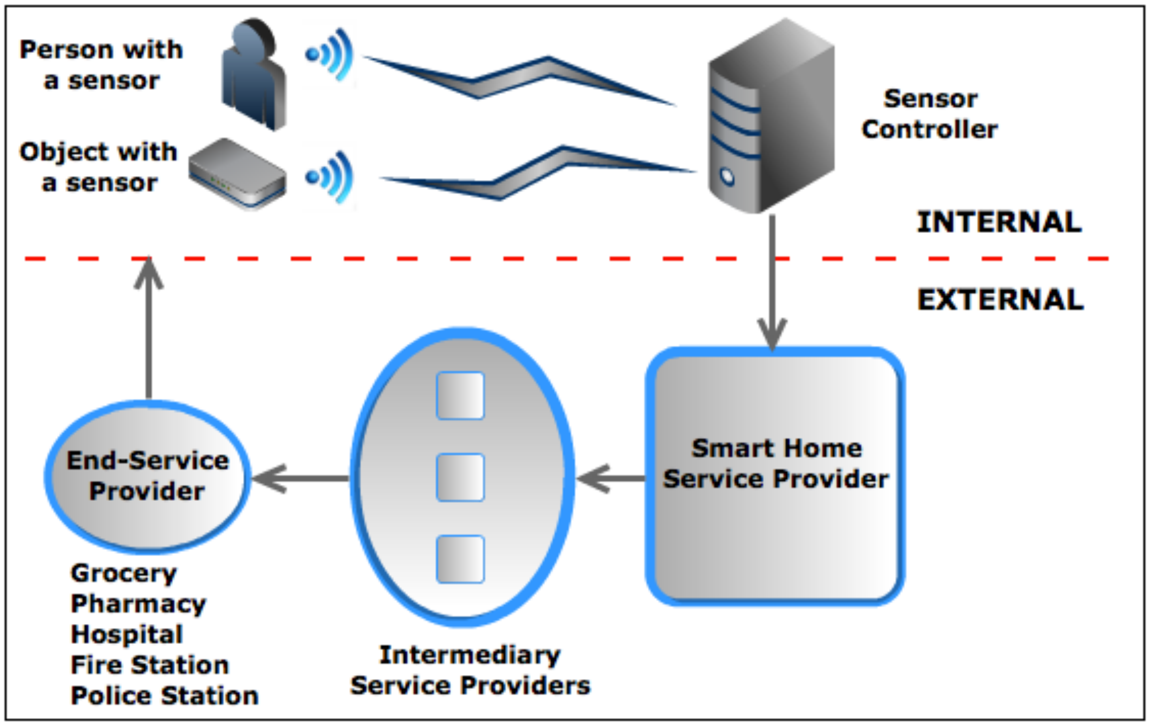


Fig. 1 Information Flow in a Smart Home System

Fig. 1 shows the information flow in a smart home scenario. A smart home system includes the information being generated and shared, IT systems - hardware, software, communication networks and their interactions that together enable the flow of information to automate activities and provide services. The components of a smart home are distributed internally in the home and externally over different service providers. The internal components include Sensors linked to persons or objects that collect and generate information and send it to the Sensor Controller. The information from various sensors is stored at Sensor Controller that dispatches the relevant information to the Smart Home Service Provider. This information is used by the Smart Home Service Provider to take appropriate actions, for e.g. submit service requests to specific service providers, forward healthcare information to healthcare providers, notify emergency to emergency services etc. At each stage in the information flow, different entities interact by using the available information, generating new information and sharing it further. This information is very sensitive as it can include all the information related to a person such as Personally Identifiable Information (PII), healthcare information, personal preferences, usage patterns, lifestyle information etc. Any information leakage, unauthorized or accidental disclosure can result in serious consequences. It can lead to huge losses and can be threatening to an individual's reputation. For instance, information leakage about a person's medical condition can result in losing her job; information about preferences and usage patterns can result in targeted advertisements and crime. This can have detrimental effects on their interactions with service providers. Their mistrust can lead to the withholding of information; the disclosure of misleading information to their health care providers; or avoidance of the certain services. Further service providers cache all the information. This can result in information gathering and aggregation leading to the creation of individual profiles. This information can be sold to advertisers; can be disclosed and used against individuals in subpoenas. Service providers can leak information during hacking attacks, insider abuse etc.

In our society, home is considered as a private location, a place of comfort, where privacy and identity of a person and her everyday activities are isolated and protected [3]. People consider home to be their trusted domain, their sphere of control for their information. An Individual has complete control over information flow, information usage and information tracking is possible in a trusted domain but when data leaves the home domain, it is very difficult to control or track data. Individuals like to keep track of their information flow to know how their information is used, with whom it is shared, and what actions are applied on it. They usually define policies to restrict access and regulate usage of their information. But once the information goes out to service providers, there is very little or no visibility and control over information usage. This information can be further outsourced to other service providers making it impossible for the information owner at the home to track or know about its current state. The magnitude of data and complexity of interactions in the information flow chain further complicates data sharing and dissemination. The threat of shared information being compromised is one of the key risks in information-based services and this risk is even more magnified when information is shared across multiple domains. Tracking the information flow and ensuring its protection throughout its lifecycle is a significant issue with cross-domain and dynamic information flows. It is very difficult to know or compare the information security controls of service providers against the information owner's policies and their capability to ensure enforcement of owner policies on the information, protect the information and level of protection actually being applied, outsourcing of information by the provider to its sub-contractors, compliance to regulatory and legal policies etc. Collecting and sharing personal information without owner's consent is a violation of privacy. Thus it is imperative that sensitive individual information in the smart home system be protected according to its owner's policies, regulatory and legal requirements. The ability to maintain the privacy and security of information will be a key determinant of the success of smart home systems. When considering privacy and security issues in a smart home system, it is not sufficient to focus only on specific internal and external components. Instead, the entire end-to-end flow of information in the system must be protected.

2. Problem Statement

The current information-sharing model has the following issues:

- No access control: Users are unable to specify access control policies for their information. There is no way for them to ensure that an authenticated entity is accessing only the information it is authorized for.
- No sharing control: Users are unable to specify policies to control sharing of their information once it leaves their trusted domain (home). There is no way for them to know with whom their information is shared.
- Loss of control: Information and services are both located with Service Providers. They handle the information as well as access control rules, security policies and their enforcement. Users have to rely on the provider to ensure data security and privacy, resource availability, monitoring of services and resources.
- Lack of trust: Trusting a third party (service provider) requires taking risks. Basically trust and risk are opposite sides of the same coin. Some monitoring or auditing capabilities are required to increase the level of trust.
- Information gathering/aggregation: Sensitive information related to a person such as Personally Identifiable Information (PII), healthcare information, personal preferences, usage patterns, lifestyle information etc is cached and stored by service providers to provide services. This information collected and stored over time can result in information gathering and aggregation leading to the creation of individual profiles.
- Information selling to advertisers: Information about user preferences and usage patterns can be sold to advertisers and can result in targeted advertisements.
- Information disclosures for Subpoenas: Information stored by service providers can be disclosed and used against individuals in subpoenas.
- Hacking attacks: Service providers can leak information during hacking attacks.
- Insider abuse: There are several entities within a service provider that have access to the information and can use the information. This can result in insider abuse leading to privacy violation.

Any non-trivial information-based application requires protecting confidentiality of disseminated data. There are several scenarios where data have to be securely shared and disseminated such as following:

- Cloud computing: Cloud computing allows service providers for using infrastructure and services offered by cloud service providers. There are many scenarios in which data is shared and disseminated across cloud users and services. For instance, the scenario of mobile cloud in which the data or computation are outsourced to cloud. Organizations and users that deploy their applications on a cloud infrastructure risk having their most valuable business and personal information disclosed to successful attackers who use the same cloud infrastructure.
- Digital Rights Management (DRM): Digital rights management blocks access to digital content (e.g., documents, movies, songs etc.) by entities not authorized by content providers. Content providers sell their digital content. Unauthorized use of digital content would be a loss for them.
- Digital Supply Chain Management: Organizations and their partners collaborate together by sharing data through supply chain management system to develop a product or service. The data in supply chain may contain sensitive information such as trade secrets, intellectual property, private organizational or personal information.

- Pervasive Healthcare Monitoring: Medical sensors and pervasive medical devices generate healthcare information about patients that is disseminated and accessed by many healthcare providers. The availability of this information to a healthcare provider could be critical for saving the life of a patient. However, allowing even inadvertent dissemination of this information to a party that may be non-authorized (such as a news company) may be damaging to the patient.
- Smart Home: A Smart Home is equipped with information-based services/devices/sensors to automate various home activities, provide seamless services, ease of access to the information that enhances occupants' quality of life. Sensitive personal information is shared with the smart home service providers in order to automate activities and provide services.
- Smart Grid: A Smart Grid allows energy consumers to buy energy from any power company, store energy locally for future use, produce energy, and sell energy back to power companies. The distribution of energy is based on the information collected from different partners (including customers). Unauthorized modification of information exchanged by the parties (i.e., falsifying the information) or dissemination of the information to inappropriate parties could disrupt the distribution of energy to customers.
- Data sharing in Military scenario: Data sharing is necessary in security challenging environments to accomplish critical missions such as communication between Aerial Vehicles for command and control.
- Service Oriented Architecture (SOA): Current business applications use web services extensively. In SOA, an entity (web service or user) can use services provided by its partners (service providers). For instance, a publisher can use fulfillment web services from Amazon.com. Companies might not use web services if confidentiality of data exchanged between the applications and web services is not protected.

Research Problem: The main research problem is how to securely share sensitive data, minimize unnecessary disclosure and protect them throughout their lifecycle?

3. Related Work

The secure data sharing solutions can be classified into three main categories based on their mechanism of policy enforcement. These are as following:

Solutions with policy enforcement at source

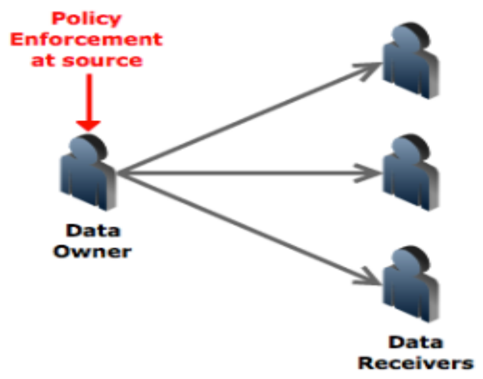


Fig. 2 Policy enforcement at source

Fig. 2 shows a typical scenario of policy enforcement at the source. This is the traditional and most common approach used in data sharing solutions. Data owner has an application on her host, which is responsible for the enforcement of her policies. The solutions are usually based on the cryptographic interactive protocols. For example Servers employ user authentication before providing a service or sharing data with their users. The problem with such solutions is that they require a lot of exchange of messages and are not scalable in all the scenarios. The source becomes the bottleneck.

Solutions with policy enforcement in the middle

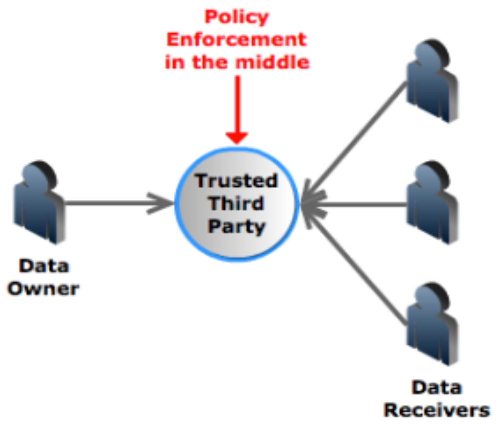


Fig. 3 Policy enforcement in the middle

Fig. 3 shows a typical scenario of policy enforcement in the middle. Such approaches use a Trusted Third Party (TTP) as a broker between data senders and data receivers. The owner sends data and policies to the TTP, which is responsible for the enforcement of policies on data. Authorized data receivers can request data from TTP. For instance, a Publish/Subscribe model is based on this approach. The subscribers subscribe to the topics of interest at the TTP and receive the respective information after the publisher publishes it at the TTP. This is a very popular model in Distributed Systems such as Virtual Synchrony. But there are many problems with this approach. There are trust issues with TTP. There is a loss of control over information published to TTP. A TTP is a single point of attack and failure and can leak information in case of hacking attacks or insider abuse. Further, TTP can cache and store the information. They can aggregate information, sell information and disclose information in case of subpoenas.

Solutions with policy enforcement at the recevier

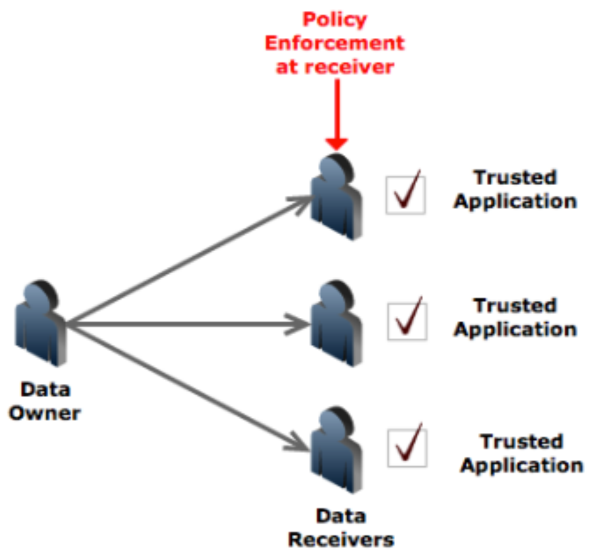


Fig. 4 Policy enforcement at the destination

Fig. 4 shows a typical scenario of policy enforcement at the receiver. In such approaches, there is a trusted hardware or software present on the receiving host, which is responsible for policy enforcement. The security policies associated with the data are either predefined and known in advance or defined during data dissemination. The policies are enforced at the host after the data is received. For instance document-based solutions from Adobe, Microsoft use this approach to authenticate users before allowing access to the encrypted documents. Digital Rights Management solutions also use this approach. The main problem with this approach is that the receiving hosts should be known in advance and should have the trusted component like a trusted application, a trusted processor etc already setup. This approach cannot be used for untrusted or unknown hosts and the distribution of trusted components to the receiving hosts is another issue.

Existing Solutions and their issues

Various solutions have been proposed for protecting confidentiality of sensitive data. Park, Sandhu, and Schifalacqua [4] present a taxonomy of architectures for managing access control and dissemination control for sensitive data and digital content. The classification is based on using or not using metadata for access and dissemination control, using or not using trusted components located on the hosts, and the dissemination style. Several proposed solutions assume that entities receiving data are already known. In such situations, the owner can use an encryption mechanism to encrypt data with keys in such a way that each entity can decrypt data that it is allowed to access. For instance, Digibox [5] protects confidentiality of sensitive data using multiple keys. Digibox is a cryptographically protected container with data and controls that enforces rights to these data. To access the data, an entity must buy a required decryption key. There are other solutions that address the problem when the data receiving entities are not known in advance. Montero et al. [6] propose a technique for enforcing confidentiality protection during dissemination of sensitive data using sticky policies. Sticky policies make data and their policies inseparable so that an attacker cannot access the data without satisfying their policies. They identify two approaches for associating data with their related policies: (i) using identifier-based encryption, and (ii) using a TPM. In the first approach, data are sent from their owner to a receiver who may disseminate them further. The owner constructs the encryption key for these data using the receiver's identifier, constraints, and conditions defined by the data access policies. The owner also encrypts data with the constructed key, and sends data and their associated data access policies to the receiver. The receiver provides its credentials and data access policies to a trusted authority (TA). The TA computes the decryption key using the receiver's credentials and the sticky policies if they are valid. Then, the TA sends the decryption key to the receiver. The second approach uses a TPM to enforce data access policies associated with a given data set. A Trusted Platform Module (TPM) is a security hardware component. It can facilitate numerous important security capabilities, such as authentication, hardware-based encryption, digital signing, secure key storage, and attestation of software installed on hardware. For encryption and signing, a TPM uses keys stored in it and not accessible to any other entity (the host that uses the TPM is excluded as well) [7]. A TPM enables building trustworthiness. For instance, it can be used to prevent tampering with software through exploiting the chain of trust [8]. The main issue with this approach is that it doesn't work if the receiving host has no TPM. Casassa Mont et al. [9] propose a solution for document dissemination when a document owner plans to send it to a set of recipients. He generates a symmetric key SK. Then, he encrypts the document using SK, encrypts SK using an encryption key based on the planned date and time of the disclosure of the document (e.g., "GMT201109151200" is an encryption key for documents to be disclosed at 12:00 a.m. GMT on Sep. 15, 2011). Next, the encrypted document is distributed to the receivers. The solution uses a time vault service, which continually generates and publishes decryption keys associated with the current date and time (obtained by a trusted clock). A receiver can open a document only with a decryption key DK, provided to the document receiver not with the document but later, at the document disclosure time. The receiver gets DK from the time vault service, and uses it to decrypt the symmetric key SK. Next, the receiver decrypts the document using SK.

4. Proposed Solution

Current solutions for protecting data consider data as passive entities that are unable to protect themselves. They require another active and trusted entity to protect them – a trusted processor, a trusted memory module, a trusted application or a trusted third party. We challenge this assumption by proposing an approach that transforms passive data into an active entity.

Research Goal: The main goal of this research is to propose a solution to control data disclosure and minimize the risk of unauthorized disclosure.

We propose an approach that transforms passive data into an active entity by encapsulating them within Managed Information Objects (MIOs) or Active Bundles (ABs) and provides better data security, protection capabilities and enhanced dissemination control.

Managed Information Object (MIO)

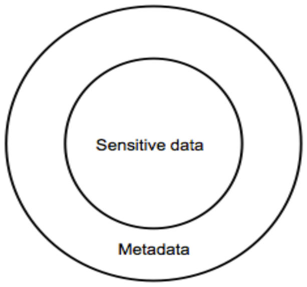


Fig. 5 Basic structure of MIO

A Managed Information Object (MIO) [10, 11] consists of data/content in its payload and metadata that describes the content and specifies various policies about content usage and protection as illustrated in fig. 5. Metadata can have different types of policies, for e.g., security policies such as access control, dissemination control, integrity control, leakage control, audit control etc and performance policies such as data discovery, data mining, quality of service etc [10]. Attaching policies to the data provides capabilities to MIO to control the behavior of data in a trusted domain. MIOs have been traditionally used in networks in protocols like SNMP (Simple Network Management Protocol) for resource management such as printers or routing the content based on policies. They can be used to build overlay networks. But if MIOs are to leave the trusted domain, they need to have control over the enforcement of policies and a protection mechanism to protect them even when outside their sphere of control. The policies alone are not sufficient until and unless there is a complete control over who enforces these policies because otherwise if there is no control then the policy enforcement cannot be trusted and may lead to data leakage and privacy violation. For this purpose, we extend MIO to provide it with a protection and policy enforcement mechanism by including a Virtual Machine (VM), which protects it and gives complete control over enforcement of its policies in any domain whether trusted or unknown. Secure MIOs are realized as Active Bundles. They protect data from within instead of outside and provide the capability to protect and control any interaction with its data.

Active Bundle

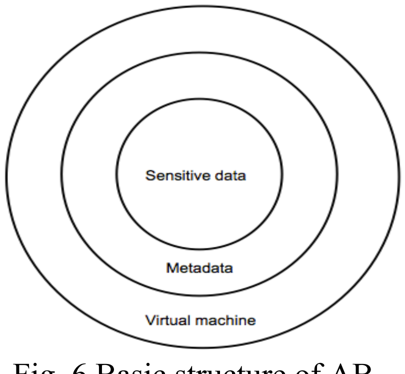


Fig. 6 Basic structure of AB

To extend MIO, we encapsulate the data and metadata with another layer that is responsible for enforcement of policies over its data and control data dissemination. The secure extended MIO is analogous to Active Bundles. An Active Bundle (AB) [12, 13] is a data protection mechanism, which can be used to protect data at various stages throughout its life cycle. An AB is an extended MIO with better security properties to protect its content. From hereon, we use the word MIO and AB interchangeably. The active bundle is a robust and an extensible scheme that can be used to disseminate data securely across multiple domains. The active bundle scheme [12, 13] is based on the AB software construct. An Active Bundle bundles together, as illustrated in fig. 6, sensitive data, metadata, and a Virtual Machine (VM) specific to the bundle.

Structure of an Active Bundle
It includes the following components:

- Sensitive data: It is the digital content that needs to be protected from privacy violations, data leaks, unauthorized dissemination, etc. The digital content can include documents, pieces of code, images, audio, video files etc. The content in the sensitive data can have several sub-elements, with different security levels and applicable policies to ascertain the content to share with a particular entity. For instance, the data sent from a UAV in an AB can contain information with multiple levels of security, for e.g., mission critical information with highest security level and terrain information with basic security level.

- Metadata: It describes the active bundle and its privacy policies. It includes (but is not limited to) the following components:
    - Provenance metadata- includes an identifier of the active bundle, identifiers of its creator and owner, the creation date, identifiers of all visited hosts, as well as identifiers of all receivers with their access times-stamps and, possibly, their update timestamps if they performed any updates of sensitive data. The history of visits and updates is kept private (it could be used only for forensic investigations by authorities that obtain a court order).
    - Integrity check metadata– includes the algorithm for checking integrity of sensitive data as well as a hash calculated by the owner.
    - Privacy policy metadata– includes access control policies for sensitive data of the bundle. They could use the required minimal host's trust level for each subset of sensitive data.
    - Dissemination control metadata– includes dissemination policies specifying who can disseminate the active bundle, and under what conditions (e.g., the active bundle could be disseminated at a specified time, or could be disseminated to a specific group of addressees such as employees, or a board of directors.)
    - Life duration– specifies a date and time where the sensitive data must disappear.
    - Security metadata– includes security server ID– a server for storing security-related information, e.g., encryption and decryption keys; the encryption algorithm used by the VM of the bundle; trust server id– used to validate the trust level and the role of a host at which the bundle arrived; and trust level threshold– specify the minimal trust level required by the VM to enable the active bundle.

    - Other application-dependent and context-dependent components– includes information related to semantics (the application, the context, etc.) of sensitive data of the active bundle.

- Virtual Machine (VM): The role of a VM is the enforcement of policies in order to guarantee the appropriate access control to sensitive data of the bundle (for example, disclosing to a receiver host only the portion of sensitive data that the it is entitled to access). A VM also performs three control mechanisms operations that we describe in the next subsection.

Operations of Active Bundles

The three main operations performed by an AB are as following:

1. Evaporation: After arriving at a host, an AB asks for the host's trust level. If the hosts' trust level is sufficient to allow access to all or portion of the AB's data, then the AB's privacy policy is applied. All data that the host is not allowed to access (as specified in the privacy policy) might be "evaporated."
2. Apoptosis: An AB may realize that its security or privacy is about to be compromised, e.g., it may discover that its self-integrity check fails, or the trust level of its recipient host is too low. In response, the AB may choose to apoptosize (i.e., perform a clean self-destruction; that is, a complete self-destruction that leaves no traces usable for an attacker. Rules for triggering an apoptosis are included in the privacy policies of AB's metadata.
3. Integrity self-check: Upon activation of an AB at a host, the AB checks its integrity using the algorithm specified in its metadata. It calculates the hash value for the AB, and compares the computed value to the value recorded within the AB's metadata. If the values differ, then AB apoptosizes.

AB Capabilities

The AB scheme protects privacy of sensitive data by (i) enforcing privacy policies specified within a bundle's metadata, (ii) activating protection mechanisms when data are tampered with, and (iii) recording audit activities involving a bundle (e.g., a transfer of the bundle from a source to a destination). In comparison to the other approaches, the AB approach is able to provide better capabilities, which are described as following:

- Quantifiable Data Dissemination: With this, the AB is able to associate a measure to the amount of data disclosed and decide to further disclose or deny more data requests based on its policies.
- Measure of Privacy with respect to each host: With this, the AB is able to associate and track the amount of data given to a particular host and a set of hosts and can relate to the effect on privacy.
- Selective Dissemination: With this, the AB is able to control the dissemination and selectively share the data based on the host's authentication level, trust level, history of interactions etc.
- Event Monitoring: With this, the AB is able to monitor and report the event activity in its lifecycle. Besides auditing, this is very useful especially in reporting a malicious activity and helps in the dynamic evaluation of hosts.
- Contextual Dissemination: With this, the AB can become context aware, so in case of a particular context like emergency, it can respond accordingly.

- Dynamic Metadata Adjustment: With this, the AB is able to update its policies, based on a context, host, history of interactions, trust level etc.

AB Implementation
Unlike other solutions, the AB scheme does not require a client application on the destination host to execute its code. It can work like an applet, a jar file or a mobile agent. The AB scheme protects sensitive data as long as they are accessed through an active bundle. Sensitive data or their parts are not protected once they are disseminated to any entity (since they leave the sphere of control of the scheme). An active bundle is an atomic digital content. It can be distributed using the push/pull model or incorporated as data packets in any data sharing or dissemination application. In the following section, we discuss the AB implementation using TTP.

Active Bundle using TTP
The TTPs are used for key management, trust evaluation of receiver hosts and auditing of AB operations. The AB prototype uses this approach. Various servers are setup so that Active bundle can communicate, send and receive information. There are three servers:
- Trust server – To evaluate the trust level of the destination host.
- Security Server – To keep track of AB cryptographic information e.g. private keys.
- Audit Server – For auditing purposes and to allow dynamic metadata updates.

The system is based on the mobile agent framework and has been implemented on top of JADE mobile agent platform [15]. A mobile agent is a software object able to perform computations on visited hosts, transport itself from one host to another, and interact with and use capabilities of visited hosts [14]. A mobile agent (analogous to an AB) contains code and carried data. Fig. 7 shows the architecture of AB prototype using JADE platform [13].

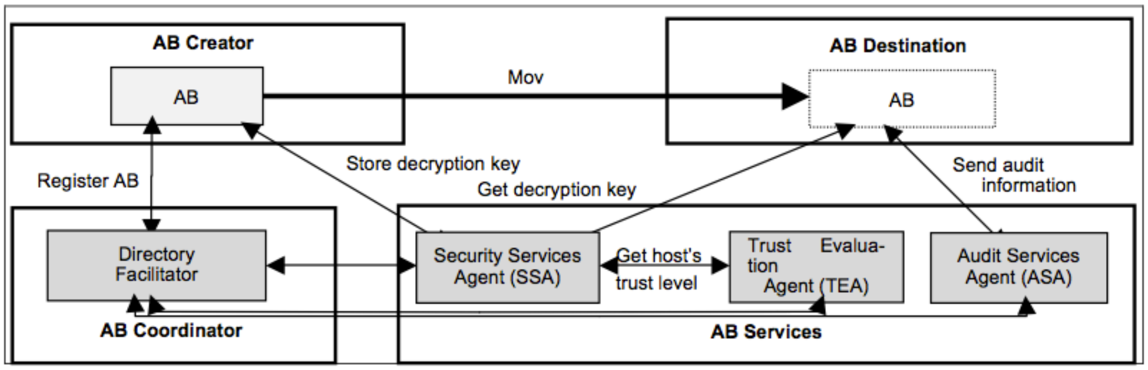


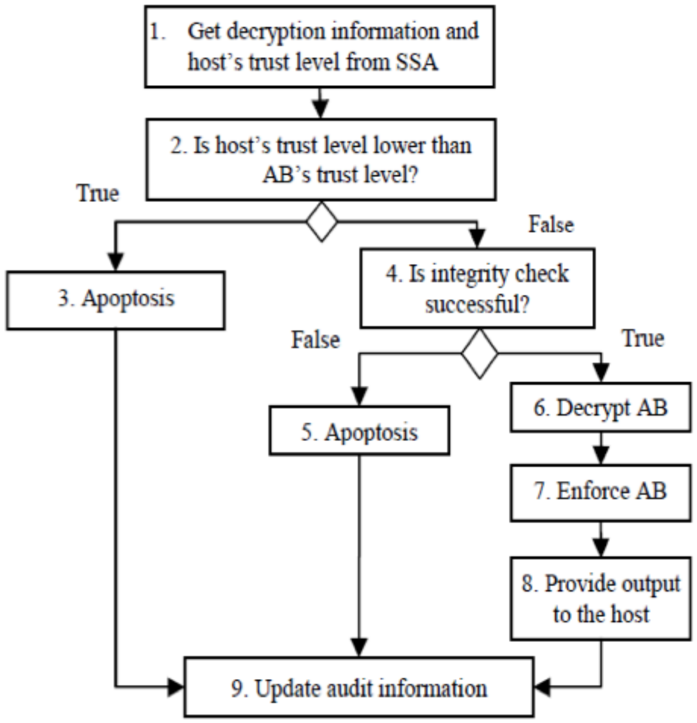
Fig. 7 UML diagram for the AB prototype using TTP

Lifecycle of Active Bundles
Below we provide a description of the lifecycle of an active bundle (AB).

Initialization of an AB: An owner of sensitive data constructs an AB by putting together data, metadata (including access control and dissemination control metadata), and adding a virtual machine. After this stage, the AB becomes an active entity (since it has its own virtual machine) that can perform the remaining steps of this algorithm.

Building an AB: The steps taken in the process of building an AB are as follows:

1. The AB gets two pairs of public/private keys from a Security Service Agent (SSA) (trusted third party), where the first pair of keys is used for encrypting the AB and the second pair of keys is used for signing/verifying the signature of sensitive data included in the AB. The reason for having two key pairs is to prevent attackers from modifying AB's sensitive data and signing it again with the public key of the data owner.
2. The AB sends a request to SSA asking it to record the AB's security information. The AB's identity data includes its name, a decryption key, and the trust level that a host must satisfy to use the AB. The goal is to keep the decryption keys and other auxiliary data for ABs in a trusted location. The decryption keys are given only to hosts that are eligible to access the AB.
3. The AB computes a hash value for sensitive data and signs them using the signature key. The signature certifies that sensitive data is from its owner.
4. The AB encrypts sensitive data using the encryption key.


1. Get decryption information and host's trust level from SSA
2. Is host's trust level lower than AB's trust level?
True
False
3. Apoptosis
4. Is integrity check successful?
False
True
5. Apoptosis
6. Decrypt AB
7. Enforce AB
8. Provide output to the host
9. Update audit information
Fig. 8 UML diagram for enabling an AB

Enabling an AB: After arriving at the destination host AB enables itself. The steps of the enabling algorithm seen in fig. 8 are as following [13]:

- Step 1: AB sends a request to SSA asking for the security information on AB and the host's trust level.
- Step 2: AB checks if the host's trust level is lower than the minimal trust level required for AB access. If so, the AB apoptosizes (i.e. completely destroys itself) (executes Step 3); otherwise, it executes Step 4.
- Step 4: AB checks integrity of its sensitive data. It computes the hash value for sensitive data and it verifies the AB's signed hash value by comparing it to the

computed hash value. If verification fails, AB apoptosizes (Step 5); otherwise, the AB decrypts the data (Step 6).
- Step 7: AB enforces its privacy policies.
- Step 8: AB provides the output to the host.
- Step 9: AB sends audit information to an Audit Services Agent. This information includes AB's name, the host's identity, and the name of the event being audited (the move to the host).

5. Selected Research Problems

In this section, we identify the issues with current AB scheme and implementation that are described as following:

- The current AB scheme relies on TTP for key management and trust evaluation. This brings all the TTP related issues to the AB scheme. TTP is a single point of failure, and it requires availability of network connection between the AB and TTP. These issues can limit the use of the solution (e.g., in the case when communication between AB and TTP can't be established).
- The current AB scheme uses conditional expressions to simulate policies. There is no discussion on policy definition for metadata.
- Adding metadata and VM increases the size of the data to be sent in the AB scheme. There is no performance evaluation of AB scheme.
- The VM in the current AB scheme uses integrity checks and apoptosis for protection against malicious hosts. But it is prone to attacks from the host executing the VM.
- The current AB prototype doesn't implement selective dissemination or evaporation.
- The current AB prototype simulates a trust evaluation service and doesn't discuss the trust evaluation mechanism.
- The current AB prototype is a general proof of concept of the AB scheme. It is not application specific.

6. Current and Future Work

In this section, we identify the required work. Our research is focused on improving the AB scheme, enhancing its capabilities, implementing the scheme in an application specific scenario and performance evaluation of the scheme in that scenario. The required work can be organized into the following categories:

AB Approach

The current AB approach uses TTP. We would like to have active bundles that do not rely on TTP. The most dangerous situation occurs when a host authorized by a bundle's privacy policy to access some (even all) its data becomes a host that is not authorized to access these data. If the host obtained decryption keys when it was still authorized, it can keep accessing bundle's data after it lost its authorization. Future work with the active bundles will focus on decreasing their reliance on a third party for security related tasks, to make them more self-protected entities. The current AB scheme doesn't provide a way for selective and controlled dissemination and evaporation. The proposed idea is to organize data into separate items. Each item can be assigned an encryption key and can be encrypted with this key. Each host can access only the data item that it is allowed to access because the decryption keys are only for decrypting specific items. Next we can use Shamir's [16] threshold secret sharing technique to split each of the decryption keys into N shares. The technique uses a threshold t that defines the minimum number of (key) shares for a specific decryption key required to reconstruct that decryption key. The reconstructed key can then be used to decrypt the respective item in the data. The decryption key shares can be stored in a distributed hash table (DHT) system (such as Vuze), where each share is stored at a separate location. A property of DHT is that data are stored (using the tuple <index, value>) redundantly on a set of nodes. The problem with using a public DHT such as Vuze is that the redundant key shares stored in a DHT expire and are removed after a specific time. After the timeout, it is not possible to reconstruct the key, which means that data become useless after a timeout since they can no longer be accessed after their key shares disappear. Another problem is that the key shares may be stored in a geographically distributed large scale DHT which makes it almost impossible to inference about the location of key shares in a reasonable time but still any information leakage about them can result in correlation and inference attacks. To solve these problems, we can use a private DHT or a hybrid DHT (combination of private and public DHTs).

Policy Definition

The current AB approach simulates but does not include definition and enforcement of privacy policies. We plan to extend AB approach with a formal language for specifying privacy policies, and with mechanisms to enforce the policies. A widely used method to specify privacy policies is to model policies using policy languages, such as OASIS eXtensible Access Control Markup Language (XACML) [17]. Privacy policy modeling has evolved to include context-aware rules and temporal logic rules. Another method to specify privacy policies is to use Role Based Access Control (RBAC) [18]. RBAC models assign users' access permissions for resources based on users' roles. The limitation of this approach is that its privacy policies cannot consider history of accesses to data. We have already extended the prototype with simple XML based policies to have some kind of policies instead of simulation. Below is a sample XML constructed for specifying sensitive data using custom defined DTD.

```
<data type="target">
<property name="color">blue</property>
<property name="coordinates">location</property>
<property name="type">type</property>
</data>
```

In this simple metadata XML based on the custom defined DTD, the properties 'coordinates' and 'type' need high security levels of access in comparison to the property 'color'. The VM is responsible to enforce correct policy on the particular content and disseminate the expected content as output.

```
<policy>
<metaData type="Sensitivity">
<data type="target" property="color" value="1" />
<data type="target" property="coordinates" value="2" />
```

```xml
<data type="target" property="type" value="3" />
</metaData>
<metaData type="AccessControl">
</metaData>
<metaData type="Dissemination">
</metaData>
<metaData type="Trust">0</metaData>
<metaData type="Role">Role</metaData>
</policy>
```

In order to provide different access levels for sensitive data as per policies, the above-mentioned idea of splitting data into separate items and encrypting each item with a different key can be used. This helps in providing better confidentiality. With this, the AB is able to provide extended security features like selective and quantifiable dissemination.

Protection against malicious hosts

The malicious hosts can attack and alter the AB VM to gain unauthorized access to bundle's data. They can also deny the VM execution. In this case, its data are not disclosed to unauthorized entities but legitimate entities are denied access to these data. The main challenge in implementing the AB's VM is assuring that a visited host executes the AB's VM code faithfully and correctly. We believe that using a TPM can help in solving some issues.

TPM is a security hardware component. It can facilitate numerous important capabilities, such as authentication, hardware-based encryption, digital signing, secure key storage, and attestation of software installed on hardware. For encryption and signing, TPM uses keys stored in a protected hardware storage [7]. TPM enables building trustworthy computing relationships. For instance, it can be used to prevent tampering software from exploiting the chain of trust [8]. For building a chain of trust, TPM evaluates a hash function for the host's BIOS, and compares this hash value to a value stored by the TPM. If both values are equal, then the BIOS can be trusted (else, it can not). If the BIOS is trustworthy, it can in turn be used to determine trustworthiness of the operating system (OS) loader. If the OS loader is trustworthy, it can in turn be used to determine trustworthiness of the OS [8]. We assume that an operating system (OS) certified by a TPM executes programs correctly. But it may be wrongly assumed that using a TPM assures secure execution of any code. In fact, it only helps in assuring secure code execution for code trusted by the owner of the TPM-equipped host. It should be remembered that TPM will faithfully execute ("trust") any software, including software with bugs or malware. Similarly, it will trust software that does not enforce privacy policies correctly. TPM does not prevent its host from tampering with host's own policy enforcement programs. In such a way, the host might successfully attack sensitive data.

We are investigating more approaches. One approach taken by previous research ([19], [20]) in secret program execution is homomorphic encryption, which operates with encrypted functions obtained by homomorphic transformations. However this approach was only shown to be usable for polynomial and rational functions, which prevents them from providing a general solution to the problem of secret execution.

Another idea is to intertwine or couple the code and data together in such a way that it is impossible to get access to the data. Basically data is hidden within the code itself. Such code can further undergo code scrambling to make it incomprehensible. Another popular approach for preventing malicious program analysis is code obfuscation. Obfuscation refers to hiding data and real program code in the plaintext form within a scrambled code [21]. The idea is to use program obfuscation to develop means to scramble code so that it still works, but provably cannot be reverse engineered [22]. A code obfuscator is a tool, which repeatedly applies semantics-preserving code transformations to a program [23]. The obfuscator tries to make a program as incomprehensible as possible in order to protect it from being reverse-engineered or to protect a secret stored in the program from being discovered. Typically, code obfuscating transformations use the following operations: fold/flatten (turn a d- dimensional construct into d+1 or d-1 dimensional ones), split/merge (turn a compound construct into two constructs or merge two constructs into one), box/unbox (add or remove a layer of abstraction), ref/deref (add or remove a level of indirection), reorder (swap two adjacent constructs), and rename (assign a new name to a labeled construct). While obfuscation makes a program more difficult to analyze, it comes at a cost: The obfuscated programs usually have greater complexity than their original, optimized counterparts, which causes worse performance. Therefore, the best practice for providing code confidentiality would be to use obfuscation sparingly, only in parts of a program, which include content that must be protected against leakage. Another obfuscation approach is based on using non-decomposable concurrent programming techniques. We propose searching for efficient algorithms able to add “foolproof” dependencies among two (or more) component programs (the original and padding programs) via branching statements. Although the active bundle scheme ensures initial code integrity, it does not offer a mechanism to protect against dynamic changes in the code during execution. Code obfuscation makes it harder for attackers to modify code in a way to achieve malicious goals. Therefore we consider the obfuscation approach for protecting program privacy. Several open-source Java obfuscation tools such as JObfuscator [24] are available, which can be used to obfuscate the AB code.

There are still other approaches that can be used like Polymorphic encryption code. The polymorphic code [25] changes itself each time it runs but the function of the code (its semantics) don't change at all. This technique can be used to hide the code and protect its execution. The main code is encrypted and a decryption function is added to the code. When the code executes, the decryption function reads the main encrypted code and decrypts it before executing it in turn. To gain polymorphic behavior, the encryptor/decryptor function pair are mutated with each copy of the code. This allows different versions of same code while all function the same semantically. Such techniques make it difficult for the malicious host to recognize the VM algorithm and code because it constantly mutates by rewriting the unencrypted decryption engine (and the resulting encrypted payload) each time it runs.

AB Application specific Development and Evaluation

We have previously proposed the use of active bundles for identity management in the cloud [26, 27] and secure data sharing in a peer-to-peer network of AVs [28] which enables selective data disclosure based on the trustworthiness of the platform. Initially, we presented Smart Home systems as a motivation. In the future, we plan to adapt AB approach and develop experiments for protecting confidentiality of data in such applications. Further we plan to measure performance of AB scheme in different deployments setups. We will compare the size of an active bundle with data size in other approaches and will also measure how the number of created active bundles affects performance of the whole system.

References

[1] Y. Gall, A. Lee and A. Kapadia, "PlexC: A Policy language for exposure control," in proceedings of SACMAT 2012.

[2] Fdslkfjlsfjk S. Helal, W. Mann, H. El-Zabadani, J. King, Y. Kaddoura, & E. Jansen, "The gator tech smart house: A programmable pervasive space," Computer, 38(3), 50-60, 2005.

[3] K. L. Courtney, "Privacy and senior willingness to adopt smart home information technology in residential care facilities," Methods of information in medicine, 2008.

[4] J. Park, R. Sandhu, J. Schifalacqua, "Security Architectures for Controlled Digital Information Dissemination", Proc. 16th Annual Computer Security Applications, Dec. 2000, pp.224-233.

[5] O.Sibert, D.Bernstein, and D. Van Wie, "DigiBox:Aself- Protecting Container for Information Commerce," Proc. USENIX Workshop on Electronic Commerce, New York, Jul. 1995 pp. 15-15.

[6] S. Montero, P. Díaz, I. Aedo, J. M. Dodero, "Towards Accountable Management of Identity and Privacy: Sticky Policies and Enforceable Tracing Services," Proc. of the 14th International Workshop on Database and Expert Systems Applications (DEXA), Prague, Czechia, Sep. 2003, pp. 377–382.

[7] Trusted Computing Group, "Cloud Computing and Security – A Natural Match," April 2010. Online at: http://www.trustedcomputinggroup.org/resources/cloud_computing_and_security__a_natural_match

[8] B. Parno, "Trust Extension as a Mechanism for Secure Code Execution on Commodity Computers," Ph.D. Thesis, Carnegie Mellon University, Pittsburgh, PA, May 2010.

[9] M. Casassa Mont, K. Harrison, M. Sadler, “The HP Time Vault Service: Innovating the Way Confidential Information is Disclosed, at the Right Time,” Sep. 2002. Online at: http://www.hpl.hp.com/techreports/2002/HPL-2002-243.pdf

[10] J. P. Loyall, et al. "QoS enabled dissemination of managed information objects in a publish-subscribe-query information broker." SPIE Defense, Security, and Sensing. International Society for Optics and Photonics, 2009.

[11] Managed Object, Wikipedia. http://en.wikipedia.org/wiki/Managed_object

[12] L. ben Othmane and L. Lilien, Protecting privacy of sensitive data dissemination using active bundles," in World Congress on Privacy, Security, Trust and the Management of e-Business (CONGRESS '09), Aug. 2009, pp. 202-213.

[13] L. ben Othmane, Active bundles for protecting confidentiality of sensitive data throughout their lifecycle," Ph.D. dissertation, Western Michigan University, December 2010.

[14] D. B. Lange and M. Oshima, “Seven Good Reasons for Mobile Agent,” Communication of the ACM, vol. 42(3), Mar. 1999, pp.88- 89.

[15] F. L. Bellifemine, G. Caire, D. Greenwood, Developing Multi-Agent Systems with JADE, John Wiley & Sons Ltd, West Sussex, England, 2007.

[16] A. Shamir, “How to Share a Secret,” Communications of the ACM, vol. 22(11), 1979, pp. 612–613.

[17] eXtensible Access Control Markup Language (XACML), Version 2.0; OASIS Standard, Feb. 2005. Available at http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml.

[18] G. Karjoth, M. Schunter, and M. Waidner, “Platform for Enterprise Privacy Practices: Privacy-enabled Management of Customer Data.” Proc. of the 2nd international Conference on Privacy Enhancing Technologies, San Francisco, CA, Apr. 2002, pp. 69-84.

[19] T. Sander and C.F. Tschudin, "Protecting Mobile Agents Against Malicious Hosts," LCNS, Mobile Agent Security, pp. 44-60, 1998.

[20] M. Brenner, J. Wiebelitz, G. Voigt, and M. Smith, "Secret Program Execution in the Cloud Applying Homomorphic Encryption," in IEEE International Conference on Digital Ecosystems and Technologies, 2011.

[21] L. Carter, J. Ferrante and C. Thomborson, “Folklore Confirmed: Reducible Flow Graphs are Exponentially Larger,” Proc. of the 30th ACM Symposium on Principles of Programming Languages, New Orleans, LA, Jan. 2003, pp. 106-114.

[22] S. Hohenberger, “Lecture 18: Program Obfuscation,” accessed in Feb. 2009. Online at: www.cs.jhu.edu/~susan/600.641/scribes/lecture18.pdf

[23] K Heffner and C Collberg, "The Obfuscation Executive," Information Security, vol. 3225, pp. 428-440, 2004.

[24] JObfuscator. [Online]. http://jobfuscator.sourceforge.net/

[25] Polymorphic Code. Wikipedia. http://en.wikipedia.org/wiki/Polymorphic_code

[26] P. Angin, B. Bhargava, R. Ranchal, N. Singh, L. ben Othmane, L. Lilien, and M. Linderman, “An Entity-centric Approach for Privacy and Identity Management in Cloud Computing,” in 29th IEEE SRDS, 2010.

[27] R. Ranchal, B. Bhargava, L. Othmane, L. Lilien, A. Kim, M. Kang, and M. Linderman, “Protection of Identity Information in Cloud Computing without Trusted Third Party,” In: 29th IEEE SRDS, 2010.

[28] B. Bhargava, P. Angin, R. Ranchal, R. Sivajumar, M. Linderman, and A. Sinclair, “A Trust based approach for Secure Data Dissemination in a Mobile Peer-to-Peer Network of AVs,” IJNGC, 2012.

[29] “Microsoft's Digital Rights Management Scheme - Technical Details,” http://cryptome.org/ms-drm.htm