PLEDGE: A POLICY-BASED SECURITY PROTOCOL FOR PROTECTING CONTENT ADDRESSABLE STORAGE ARCHITECTURES

Wassim Itani Ayman Kayssi Ali Chehab

Department of Electrical and Computer Engineering
American University of Beirut
Beirut 1107 2020, Lebanon
{wgi01,ayman,chehab}@aub.edu.lb

ABSTRACT

In this paper we present PLEDGE, an efficient and scalable security ProtocoL for protecting fixed-content objects in contEnt aDdressable storaGe (CAS) architEctures. PLEDGE follows an end-to-end policy-driven security approach to secure the confidentiality, integrity, and authenticity of fixed-content entities over the enterprise network links and in the nodes of the CAS device. It utilizes a customizable and configurable extensible mark-up language (XML) security policy to provide flexible, multi-level, and fine-grained encryption and hashing methodologies to fixed content CAS entities. PLEDGE secures data objects based on their content and sensitivity and highly overcomes the performance of bulk and raw encryption protocols such as the Secure Socket Layer (SSL) and the Transport Layer Security (TLS) protocols. Moreover, PLEDGE transparently stores sensitive objects encrypted (partially or totally) in the CAS storage nodes without affecting the CAS storage system operation or performance and takes into consideration the processing load, computing power, and memory capabilities of the client devices which may be constrained by limited processing power, memory resources, or network connectivity. PLEDGE complies with regulations such as the Health Insurance Portability and Accountability Act (HIPAA) requirements and the SEC Rule 17a-4 financial standards. The protocol is implemented in a real CAS network using an EMC Centera backend storage device. The application secured by PLEDGE in the sample implementation is an X-Ray radiography scanning system in a healthcare network environment. The experimental test bed implementation conducted shows a speedup factor of three over raw encryption security mechanisms.

KEYWORDS

Security, Content-addressable storage security, Policy-driven security, Customizable security.

1. INTRODUCTION

Content-addressable storage (CAS) is a novel storage methodology targeting the efficient online storage and retrieval of fixed-content entities in enterprise network environments. Fixed-content entities refer to objects that should be stored in a fixed format and whose content should be conserved intact for extended periods of time. This is either due to the business value of these entities, their future significance, or due to industrial regulations and policies that enforce the authentic storage of such entities. Of these regulations one can mention the Health Insurance Portability and Accountability Act (HIPAA) [5] for securing medical records and patient information against theft, disclosure, or modification, and the Gramm-Leach-Bliley Act [16] for ensuring the confidentiality and integrity of financial records and banking information for any institution providing a financial service. In fact, if one looks at the information life cycle of different types of transactional data, it becomes clear that all data eventually end as fixed content. Entities such as emails and attachments, X-Ray images, CAT scans, CAD/CAM

drawings, financial records, legal contracts, database backups, etc. may start transactional and may involve some modification, but in their final stages they all end up as fixed-content.

The main problem facing the storage of fixed content data is that this content is conventionally archived on backup tapes and/or optical disks due to the economy of such storage media. This backup technology, however, results in a very slow interaction with the stored entities, and to an immense limitation in the accessibility and usefulness of the stored information, which contradicts regulatory policies, auditing obligations, and customer requirements that demand the continuous and online access to information for relatively long periods of time. To solve this problem, the CAS technology provides a relatively cheap storage solution that facilitates very fast online network accessibility to fixed-content entities with the guarantee of data integrity against any form of modification or fabrication which complies with regulatory requirements and customer needs. CAS refers to the fixed content entity by a unique global identifier which is the hash value of the contents of this object, rather than using the traditional way of accessing the object by its name in a physical location.

CAS storage nodes are used to store sensitive and critical data. The results of clinical trials such as X-Ray images, MRI scans and CAT scans; and financial documents such as check images and insurance photos are few examples of such sensitive data. To prevent any loss of confidence, reputation, and trust among clients and to avoid legal or civil fines and penalties, companies and institutions must exert considerable effort and take substantial precautions to protect the confidentiality and integrity of their fixed content entities. When designing CAS security, one should consider two main objectives: the first one is to safeguard the confidentiality, integrity, and authenticity of data entities stored in the CAS storage nodes, and the second objective is to protect the confidentiality and consistency of the data travelling over the network links in the CAS enterprise network. Both objectives should be given exceptional attention.

The operation of today's CAS enterprise applications over different network infrastructures using client devices and terminal workstations that vary greatly in capabilities and resources makes such applications very assorted in requirements, configurations, and resources. This necessitates that the security protocols used in securing such applications be designed specifically for operation in such diverse environments. The security protocols must consider the large variety of client devices which may be severely constrained in terms of processor speed, memory resources, and storage capacity. Many hospitals today are supporting their staff with palm-sized pocket PCs and Personal Digital Assistants (PDAs) mainly operating over wireless networks for recording clinical trials and even analyzing and distributing X-Ray scans and images. This diversity makes the implementation of a unique security standard that encompasses the whole device range infeasible. A least-common denominator security solution that targets devices with limited memory and slow processors would be unfair to powerful devices and would not meet their security requirements, and in the same sense, a security solution that addresses high-end devices would neither fit nor perform efficiently on limited-resource devices. What is needed is a security protocol that can be customized and configured to perform the security operations flexibly, taking into consideration the memory capabilities and the processing power of the client device, the network latency, and the specific requirements of the enterprise CAS application. This ensures the efficient operation of the same application on a wide range of devices and enterprise networks. Moreover, this protocol must be extensible, scalable and capable of evolving to meet new challenges and to adapt to new application requirements.

In this paper we present PLEDGE, an efficient and scalable security protocol for protecting fixed-content objects in CAS architectures. PLEDGE follows an end-to-end policy-driven security approach to secure the confidentiality, integrity, and authenticity of fixed-content entities over the enterprise network links and in the nodes of the CAS device. It utilizes a customizable and configurable XML [15] security policy to provide flexible, multi-level, and

fine-grained encryption and hashing methodologies to fixed content CAS entities. PLEDGE secures data objects based on their content and sensitivity and highly overcomes the performance of bulk and raw encryption protocols such as the SSL [2] and TLS [3] protocols. Moreover, PLEDGE transparently stores sensitive objects encrypted (partially or totally) in the CAS storage nodes without affecting the CAS storage system operation or performance and takes into consideration the processing load, computing power, and memory capabilities of the client devices. By transparently offering content-based security services to CAS systems, PLEDGE lays the ground for achieving the optimal point of intersection between security and efficiency in fixed-content, high-speed storage environments. The cryptographic mechanisms provided complement and support the regulatory compliance policies by assuring the security of data in the rest and transit states.

The work presented in this paper is an extension to [1]. The additions incorporated include elaborate discussions on the operation of the core security protocol, extensive details of the security policy generation methodology, updates on the key management mechanism to enhance its resistance to device seizure attacks, and the implementation of the security protocol on a new CAS box and client platforms.

It should be noted here that the PLEDGE protocol can be tuned to secure transactional storage environments, such as file systems, relational database management systems, or directory servers. However, two chief properties give a content-driven security solution sizable importance in the CAS realm. First, the object-based structure of CAS fixed-content entities supports the content-based encryption mechanisms by facilitating the process of selecting conceptual encryption regions based on high-level specifications. This property is highly realized when dealing with imaging objects, video objects or document scans. Second, the relatively large physical size of CAS entities dramatically enhances the performance of content-based encryption operations especially when the encryption zones selected occupy relatively small fractions of the CAS object compared to the whole object size.

The rest of the paper is organized as follows: in Section 2, we review some previous work related to the security of CAS systems. In Section 3, we present the threat model assumed in this work. In Section 4, we describe the main design features of PLEDGE and its architectural components. Section 5 formulates a mathematical model for performance of the policy-based encryption used in PLEDGE. Results for a prototype PLEDGE implementation are shown in Section 6, and conclusions are presented in Section 7.

2. RELATED WORK

As stated previously, two main requirements must be satisfied to achieve the confidentiality and integrity of fixed content entities in a CAS environment. The first requirement is securing the data at rest in the CAS storage nodes, while the second is protecting the data during its marshalling over the network links in the CAS enterprise network.

2.1 Storage Security

CAS environments usually store sensitive information, such as medical records and financial documents, whose confidentiality and integrity should be guaranteed in any situation and under any circumstance. While PLEDGE is unique in using a policy-based approach, which has been successfully applied in other domains [11-14], other solutions have been proposed to enhance the performance of secure storage systems. Lakshmanan [26] proposed a design for a distributed store integrating replication and secret sharing, providing better flexibility and higher performance. The human immune system was also an inspiration for artificial immune strategies, aiming at improving the security of data storage and resolving the problem of resource sharing in a storage area network. The approach presented in [27] assigns for each user

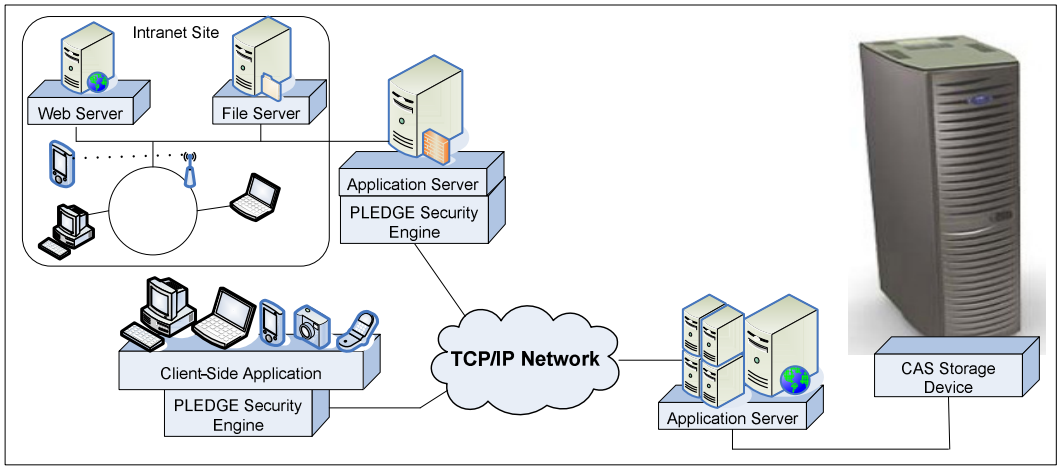

Figure 1. PLEDGE Scope and Operation

a vaccine which is pseudo-noise that is used when writing or reading the data. Another solution proposed a three stage authentication procedure for both the client and the server, to provide a virtual encryption of the data between the client and the server [28]. A new approach was presented in [29], which groups sensitive data before encrypting it, thus reducing the encryption overhead and improving performance. To prevent tampering or deleting the stored data, or to protect data in compromised systems, an internal audit of all requests was proposed in [30], in addition to keeping a copy of older versions of the data for a certain period of time. This solution is said not to affect performance, especially when data compression is used. An infrastructure for large-scale systems was proposed as OceanStore in [17]. It provides support for naming and access control, archival storage, and an update model to allow concurrent updates. This infrastructure takes into consideration security and assumes that data servers are untrustworthy, thus server-side operations are designed to be done on encrypted information.

2.2 Network Security

Most CAS solutions secure their data objects over the network links using bulk encryption protocols such as SSL and TLS. These raw security protocols impose a large performance overhead on the client due to the huge amount of data this machine should process from the typically large size (tens of megabytes and even gigabytes) of fixed-content data objects. Add to this the fact that the client machine should be dedicated for analyzing and processing these data objects (consider the X-Ray analysis client application) which makes the performance impact even higher. Moreover, SSL encrypts data objects without regard to their type or sensitivity. To SSL, network data is of one type, and there is no categorization of this network data based on content or sensitivity. Thus, when using SSL, all the fixed-content objects are encrypted by the same cryptographic key-strength, which can be unnecessary or even undesirable for some CAS applications. It is worth mentioning here that the SSL protocol provides its data security services in a point-to-point mode that protects the network communication channel from the socket on the client process to the socket on the web server process. This mode of operation makes SSL unfeasible for securing storage data entities at rest.

3. THREAT MODEL

The threat model assumed in this paper is described as follows: we have a CAS storage network consisting of 3 main components: a CAS storage device for storing fixed-content objects, a set of client devices capable of generating fixed-content objects, and an application server for interfacing the client application with the CAS box and providing the necessary middleware

functionality. The attacker in PLEDGE's threat model is capable of jeopardizing the integrity and confidentiality of network traffic by executing man-in-the-middle attacks on the network links. In other words, the attacker is supported with the necessary tools and expertise to conduct any type of sniffing, modification, and fabrication attacks on the network data. Moreover, the threat model is strong enough to incorporate the ability of the attacker to break into one of the network devices, particularly the client device and the CAS storage box.

4. PLEDGE DESIGN AND ARCHITECTURE

In this section we present the main components of the PLEDGE architecture and their interactions to provide a scalable and customizable security system. Figure 1 presents a conceptual view of the different components comprising this security architecture. A more comprehensive and functional description of each component is presented later in this section.

PLEGDE follows a policy-driven approach in securing fixed content entities over the network links and in the CAS storage nodes. The main components constituting the PLEDGE architecture are the client device, the applications server, the security engine, and the CAS storage system. The security engine is the component responsible of ensuring the confidentiality, integrity, and authenticity of the fixed data elements in the PLEDGE system. The security services supported by the PLEDGE's security engine are controlled and configured based on information present in a policy configuration file attached to each data entity generated or captured by the client device. The policy configuration file provides the primary information source that the security engine consults for taking security decisions and carrying out security-related mechanisms and procedures. A very important design goal achieved in PLEDGE is the ability to make the security system *transparent* to the CAS storage devices. The CAS storage system operation and all its interfaces with the enterprise network are not affected in any way by PLEDGE's security mechanisms and policies.

The rest of this section will be devoted to presenting the role, functionality, and interaction of the different security components building the PLEDGE architecture. Moreover, the detailed steps of the PLEDGE security protocol in storing and retrieving fixed content objects and in generating ciphering keys and other security-related parameters are presented.

4.1 The Security Policy

Every fixed content entity to be stored and secured in the CAS enterprise network is linked to a security policy file that specifies the security behaviour and operation of the PLEDGE security architecture. The most important criteria that the security policy identifies is the part (or parts) of the CAS object to be secured and the level or degree of encryption to be applied to these parts. In other words, the security policy consists of a set of conditions and rules over which the policy should be applied, and a set of actions that should be executed in the event of satisfying the policy conditions. The source information of this policy is encoded in an XML formatted file which is produced upon the CAS object creation or capturing by the client device. Utilizing a customizable security policy in the architecture enhances to a great degree the scalability, flexibility, and customization of the security system. Moreover, it facilitates the process of managing and administering the different elements composing the security architecture. This is considered very vital in today's complex and intricate enterprise systems where the process of managing and securing business data is not in any way a simple task. To facilitate the policy creation procedure, PLEDGE provides a user-friendly graphical user interface utility on the client device that facilitates the creation and management of security policy configuration files and helps the security administrator create and administer these policy files rapidly without requiring knowledge of the exact details of the policy file syntax. This assists in the construction of valid and error-free policy configuration files.

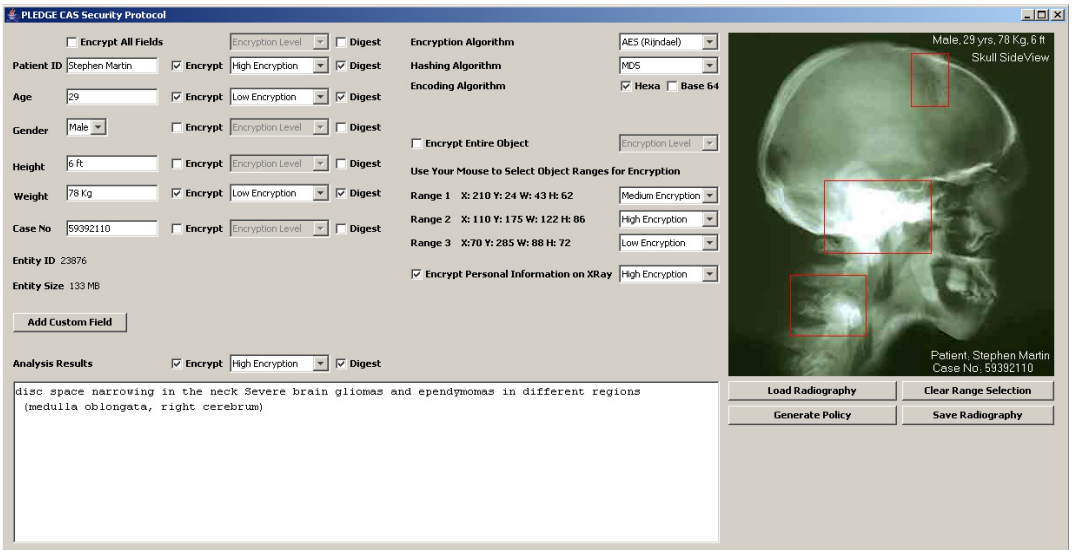

Figure 2. PLEDGE Policy Generation GUI

The policy administration utility plays a major role in checking the syntactic and semantic structure of the policy file and determining whether this policy file is compatible with the device and network resources and other security performance constraints. As will be illustrated later in the implementation and performance section, the CAS application implemented and secured by the PLEDGE protocol is an X-Ray radiology capturing system in a hospital network. The X-Ray radiology scans are captured and generated by the client device, secured by the client-side or application server security engine, and transmitted for secure storage in the CAS storage nodes.

The security policy consists of the following three main sections:

The security parameters section: this section specifies the security attributes that should be globally applied on the fixed data entities in securing their transmission and storage in the CAS environment. Some possible attributes are: the encryption algorithm, the hashing algorithm, the key exchange algorithm, and the data encoding protocol.

The metadata section: this section specifies some metadata properties that are linked to the CAS object and which could be used as searching criteria when querying the CAS storage box. The policy may specify that some or all of these metadata properties be encrypted for confidentiality issues or added to the hash chain for data integrity purposes. The policy may also indicate the encryption or security level that should be applied on the specified metadata properties. The number of encryption levels that the policy-based architecture can provide depends on the different key lengths supported by the encryption algorithm used. The following four encryption levels are provided: a ***High_Security*** level which is equivalent to 256-bit AES [7] encryption; a ***Medium_Security*** level which is equivalent to 192-bit AES encryption; a ***Low_Security*** level which is equivalent to 128-bit AES encryption, and a ***No_Security*** level which represents no encryption or security on the field. It should be noted that as the key length increases, the security of the system and the processing requirements of the algorithm increase.

The CAS entity section: this is the most crucial section in the security policy that specifies the encryption mechanisms that should be applied on the CAS object. It indicates the object ranges that should be secured in the fixed content entity and the level of security to be applied. It should be noted that the range could be the entire object. The range specification is implementation-dependent and relies on the type of application and the nature of the fixed content entity. In the sample implementation we show in this paper, we deal with X-Ray

radiography images; the range is specified in the form of rectangular regions by indicating the X and Y coordinates of the top left point and the width and height of the rectangle sides. It should be noted that the partial image encryption algorithm for securing the various regions of the X-Ray images is implementation dependent and the implementer can choose from a wide variety of algorithms targeting this subject [18-25].

In PLEGDE's sample implementation the policy is constructed using a very intuitive GUI incorporated into the medical software used for capturing the X-Ray scans. Figure 2 presents a screenshot of the policy generation tool interface. The tool GUI is Java-based and developed using the Swing API toolkit. It incorporates all the required constructs and components to generate and edit the different sections of the security policy. The left part of the frame (including the diagnosis results) produces the fields of the *metadata* section of the security policy. This includes the definition of the security strength on the different fields, and the ability to add custom metadata attributes.

The rightmost section of the GUI displays the X-Ray image on which the user can define the various regions to be encrypted by the content-based security engine. The specification of the secure rectangular boundaries is facilitated using simple mouse selection schemes. The top middle section of the GUI renders the global attributes of the *security parameters* section of the policy. This is followed by the fields of the *CAS entity* section, where the user can indicate the security level of the different regions selected on the X-Ray image.

The XML representation shown in Figure 3 is a sample security policy generated by the GUI presented in Figure 2. As clearly seen in Figure 3, the structural elements composing the XML file provide a direct mapping to the different sections of the security policy. This mapping is illustrated as follows:

- The <*security_parameters*> element represents the *security parameters* section of the policy. This element includes a specification of the global security attributes, such as the encryption, hashing, key management, and data encoding algorithms.

- The <*metafields*> XML element encloses the representation of the different metadata fields included in the *metadata* section of the security policy. This is accompanied by indicating the security properties of each metadata field using the *encrypt*, *level*, and *integrity_enforcement* XML attributes.

- The <*CAS_entity*> element represents the *CAS entity* section of the security policy. The *encrypt_all* attribute causes the content-based security engine to switch to the raw encryption mode. Although this mode is not recommended, it might be employed when the enterprise application is not performance-critical or when comparing the performance of the content-based security engine against raw encryption mechanisms.

A very important child element of the <*CAS_entity*> element is <*SecureRange*>. From its name, <*SecureRange*> specifies the image security zones and their security properties. The security zones are rectangular regions defined by the (x,y) coordinates of the top left rectangle point using the <*X*> and <*Y*> elements respectively; and by the width and height of the rectangle sides using the <*W*> and <*H*> elements respectively.

Securing specific regions in the CAS entity is considered a very vital design goal achieved in PLEDGE. For example in securing financial contracts and bank check images it is sufficient to encrypt certain parts of the document that are related to the customer identity and the monetary amount. In the X-Ray radiography scan application, the only image parts that need to be encrypted are the patient's personal information on the image and any image parts showing serious health problems such as tumours which usually occupy relatively small portions of the X-Ray scan compared to the whole image size.

```
<?xml version="1.0" encoding="UTF-8"?>
<Pledge version=1.0 platform="EMC Centera x.x">
<security_parameters>
<encryption_algorithm>Rijndael</encryption_algorithm>
<hashing_algorithm>MD5</hashing_algorithm>
<key_management>Diffie-Hellman </key_management>
<data_encoding>Hexadecimal</data_encoding>
</security_parameters>
<metafields encrypt_all="false" integrity_enforcement="false">
<metafield encrypt="true" level="high" integrity_enforcement="true">
        <name>PatientID</name>
        <value>Stephen Martin</value>
</metafield>
<metafield encrypt="true" level= "low" integrity_enforcement="true">
        <name>Age</name>
        <value>29</value>
</metafield>
<metafield encrypt="false" integrity_enforcement="false">
        <name>Gender</name>
        <value>Male</value>
</metafield>
<metafield encrypt="false"  integrity_enforcement="false">
        <name>Height</name>
        <value>6 ft</value>
</metafield>
<metafield encrypt="true"  level = "low" integrity_enforcement="true">
        <name>Weight</name>
        <value>78 Kg</value>
</metafield>
<metafield encrypt="true" level = "high" integrity_enforcement="true">
        <name>Analysis Results</name>
        <value> disc space narrowing in the neck
Severe brain gliomas and ependymomas in different regions (medulla oblongata, right cerebrum)</value>
</metafield>
<metafield encrypt="false" integrity_enforcement="true">
        <name>Case No</name>
        <value> 59392110</value>
</metafield>
</metafields>
<CAS_entity encrypt_all="false">
<EntityID> 23876</EntityID>
<EntitySize>133 MB</EntitySize>
<SecureRange id="1" encrypt="true" level = "medium" integrity_enforcement="true">
<X>210</X>
<Y>24</Y>
<W>43</W>
<H>62</H>
</SecureRange >
<SecureRange id="2" encrypt="true" level = "high" integrity_enforcement="true">
<X>110</X>
<Y>175</Y>
<W>122</W>
<H>86</H>
</SecureRange >
<SecureRange id="3" encrypt="true" level = "low" integrity_enforcement="true">
<X>70</X>
<Y>285</Y>
<W>88</W>
<H>72</H>
</SecureRange >
<Entity_Info encrypt="true" level="high"/>
</CAS_entity>
</Pledge>
```

Figure 3. PLEDGE Sample XML Security Policy

This, in addition to guaranteeing the patients rights in keeping their health status and record private and protected against any form of misuse or disclosure, contributes to decreasing the number of encryption operations and controlling their level which results in great flexibility and an overall performance improvement.

4.2 The Security Engine

The security engine is the most important component in the security architecture due to the great role it plays in providing, directly or indirectly, the essential security services needed for protecting the CAS enterprise data and securing its storage. The security engine is responsible for providing authentication and data confidentiality and integrity services. It participates in carrying out an efficient and secure key generation mechanism and ensures the storage security of the ciphering keys and other sensitive parameters on the client device/application server. The security services supported by the security engine are controlled and configured based on information present in the policy runtime process which provides the primary information source the security engine consults for taking security decisions at runtime.

The operation of the security engine is functionally divided into two main phases: The secure object storage phase and the secure object retrieval phase. In the first phase the client application takes the responsibility of capturing the CAS object θ_n, generating its corresponding security policy P, and encrypting its content based on the rules specified in the security policy. The encryption operations are also applied on the object's metadata attributes. The encrypted result is appended to the object's content address Ψ_n (generated by hashing the policy-based encryption result of θ_n) and its identification number n. The resulting message is sent authenticated with a message authentication code (MAC) to the application server. The key w to the MAC function is a key shared between the client and application server at system initialization time using public-key algorithms. The public-key algorithm used to share the symmetric key w is implementation dependent. For instance, the implementer may use an authenticated version of the *Diffie-Hellman* key management protocol [31] between the client and application server to securely agree on a shared key w, or the client can generate w at random and send it encrypted with the application server's public key. The application server can then decrypt the encrypted content with his private key to retrieve the shared secret w. Note that the public key distribution mechanism is based on the traditional public-key certificates approach. This mechanism assumes that each client device is securely embedded with the public-key certificate of the application server. The same argument applies to the generation of the shared secret X between the application server and the CAS storage device. The multi-level ciphering keys used by the content-based security engine are produced by hashing a session key EK_n with 128-bit, 192-bit, and 256-bit hash functions.

Upon receiving the client's message, the application server forwards it to the CAS storage nodes after verifying the authenticity of the MAC value. On the storage side, the CAS node stores the encrypted object and generates a system identifier Υ that virtually points to the logical storage location of the object. Whenever the CAS node is requested to retrieve a stored object, it converts the handle Υ to a physical address by employing an internal mapping process. Υ is sent authenticated to the application server to be stored in a local database.

In the secure object retrieval phase, the client application sends the content address Ψ_n to the application server. The latter queries its local database to retrieve the object's handle Υ and send it to the CAS storage device. Using Υ, the CAS nodes can determine the physical address of the object to retrieve it and send it with MAC integrity authentication to the application server. The application server receives the policy-encrypted object and sends it to the client application after verifying the authenticity of the MAC. Together with the encrypted object, the application server presents the client with the object's encrypted metadata attributes, its content address Ψ_n, and its identification number n.

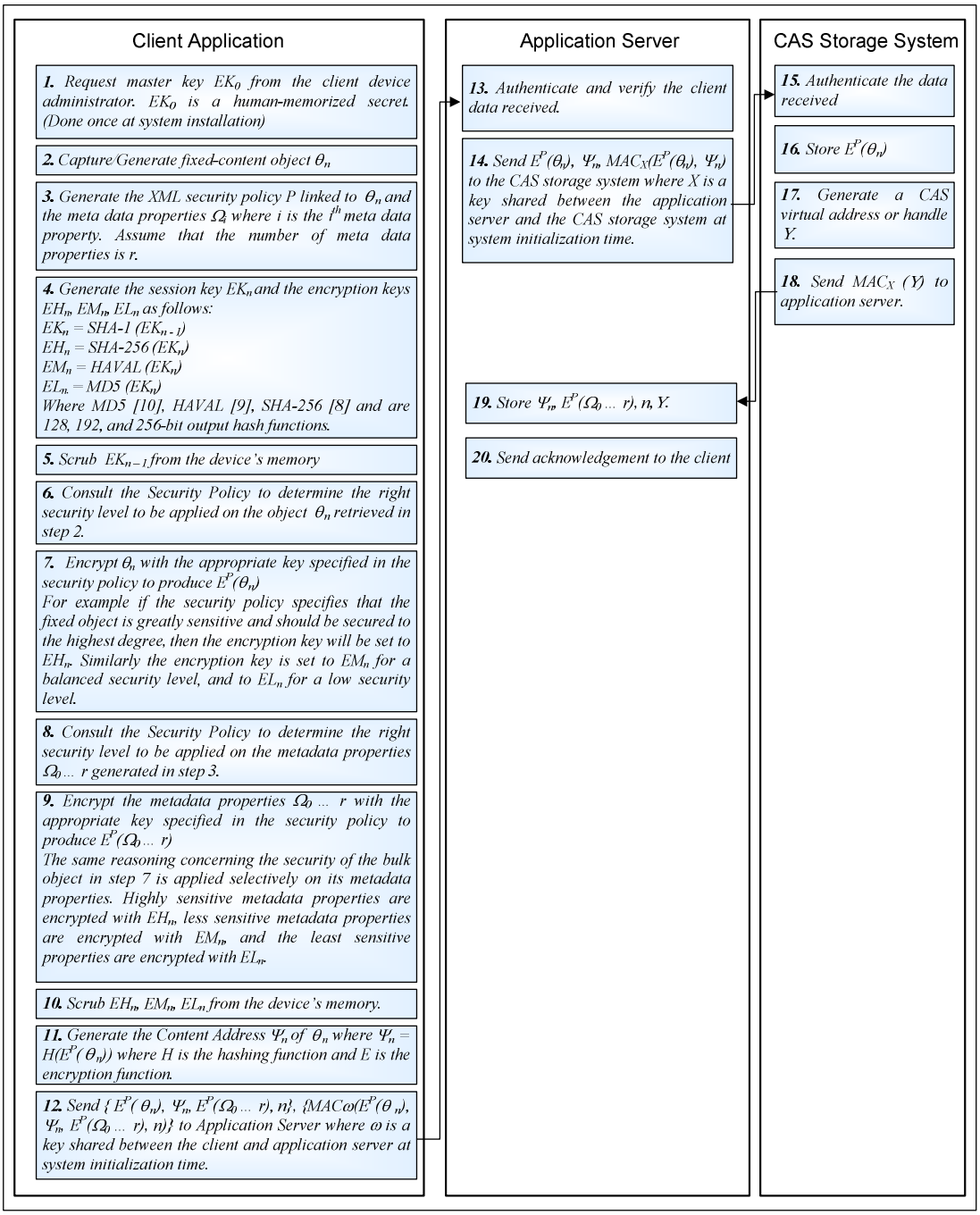

Figure 4. Secure Object Storage Protocol Steps

On the client side, the security engine decrypts the contents of the object after generating the session key EK_n and the encryption keys EH_n, EM_n, EL_n, using n and EK_0. The detailed steps of the PLEDGE security protocol in the secure storage and retrieval of fixed content objects and the functional role of the security engine in this protocol are illustrated in Figures 4 and 5 respectively.

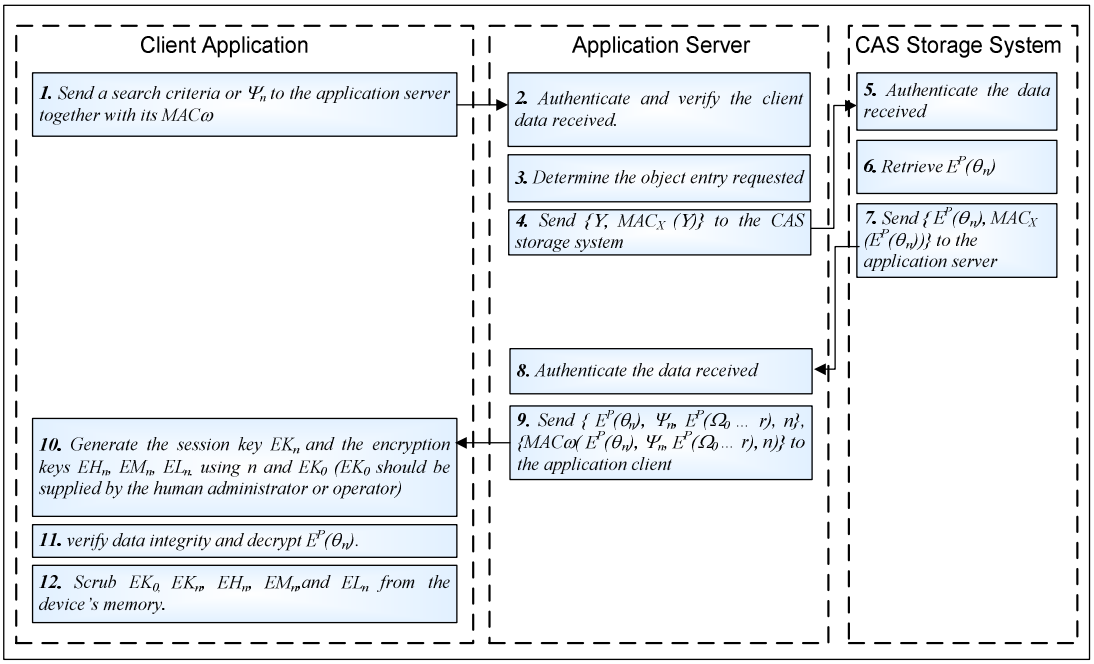

Figure 5. Secure Object Retrieval Protocol Steps

The design of the Security Engine makes it impossible for an attacker capturing the client device to disclose session keys generated before the time of the attack. If the attacker captures the device after EK_n is generated, she will only be able to retrieve EK_n from the memory of the device. Since EK_n is derived using a one-way hash function from EK_{n-1}, the attacker will not be able to deduce EK_{n-1}, and as a result the encryption keys EH_{n-1}, EM_{n-1}, and EL_{n-1} derived from EK_{n-1} are also protected. With such design, the privacy and confidentiality of the session keys are assured and the authenticity of the hash chain is guaranteed. It should be noted here that an attacker capturing the master key EK_0 would jeopardize the security of the whole model. This issue can be practically solved by setting EK_0 to a client memorized secret that is entered once at system initialization time and scrubbed from the device's memory once EK_1 is generated. EK_1 and the following session keys and their corresponding encryption keys in the hash chain are similarly swiped from memory once their encryption/decryption task is completed. Moreover, for the safety of the master key EK_0, the client should enter the memorized secret EK_0 every time a CAS object is to be decrypted (see steps 5 and 10 in Figure 4 and steps 10 and 12 in Figure 5).

As shown in Figure 1, the policy-based security architecture can be implemented to operate on the proxy server side to provide the data assurance to an entire Intranet site or Local Area Network (LAN). This scenario is suitable for providing policy-based security services to mobile devices with limited resources that are not capable of running the security engine natively.

4.3 The Policy Runtime Process

The policy runtime process is a compact, internal representation of the security policy in the memory of the client device or the application server, depending on where the PLEDGE security engine is running. This runtime process is initialized at policy generation time and is the primary source that the security engine consults for taking security decisions and carrying out security-related operations at runtime.

4.4 Policy Security

The security policy contains sensitive information that controls the various security operations in PLEDGE. Any malicious modification to this information may lead to dangerous security attacks. Thus, assuring the integrity of the policy information must be given exceptional attention. MACs and digital signatures are used to guarantee the authenticity of the policy information on the client device.

5. SYSTEM PERFORMANCE MODEL

This section presents a formal mathematical analysis of the performance of PLEDGE. Most of the equations presented in this performance analysis are conducted in a platform-neutral manner without relying on any device hardware or operating system. In the rest of this section we will use the following notation:

- N is the number of encryption levels supported by the policy-based security architecture. This depends on the number of key lengths supported by the encryption algorithm ($N = 3$ in PLEDGE's implementation).
- $T = SizeOf(\theta_n)$ is the size in bytes of the n^{th} fixed content entity.
- $E_T = SizeOf(E^P(\theta_n))$ is the number of bytes encrypted in the n^{th} fixed content entity (this is the sum of the sizes of the security ranges specified in the security policy).
- μ_i is the percentage of bytes encrypted by the i^{th} encryption level based on the specifications of the security policy

5.1 Percent Decrease in Encrypted Bytes

PDE is the percent decrease in encryption operations resulting from the use of policy-based encryption mechanisms. Compared to bulk encryption, the percent decrease in encryption operations is obtained as follows:

$$PDE = \frac{T - E_T}{T} \times 100 \qquad (1)$$

5.2 Performance Gain

We now calculate the performance gain achieved due to the policy-based security.

Defining W_i to be the number of bytes encrypted by the i^{th} encryption level, the percentage of bytes encrypted according to the i^{th} encryption level is obtained as follows:

$$\mu_i = \frac{W_i}{T} \times 100 \qquad (2)$$

Let C_i be the cost of an encryption operation using the i^{th} encryption level. This cost may be the number of CPU cycles to perform an encryption operation or the RAM footprint consumed by an encryption operation, or a function of both. It should be noted that the values of the different C_i *s* are platform-dependent but $C_1 < C_i < C_N$ since the complexity, and therefore cost, of encryption increases with the size of the encryption key. Assume that the traditional security approach of securing all the fixed content entity uses an encryption strength equivalent to C_r; we define

$$J = \sum_{i=1}^{N} \frac{\mu_i}{100} \times C_i \qquad (3)$$

The performance gain *G*, resulting from the use of policy-based security mechanisms relative to the traditional approach which encrypts all the object will be

$$G = \frac{C_i - J}{C_i} \times 100 \quad (4)$$

If the policy-based architecture uses a three-level encryption scheme (low, medium, and high) utilizing an encryption algorithm with three key sizes such as AES (128, 192, and 256 bit keys) then the percentage of bytes encrypted according to the high, medium and low encryption levels are respectively obtained as follows:

$$\mu_3 = \frac{W_3}{T} \times 100 \quad (5)$$

$$\mu_2 = \frac{W_2}{T} \times 100 \quad (6)$$

$$\mu_1 = \frac{W_1}{T} \times 100 \quad (7)$$

Assume that the traditional security approach of securing all the contents of the CAS entity uses an encryption strength equivalent to the medium encryption level that is, $C_r = C_2$, we will have:

$$J = \frac{\mu_1}{100} \times C_1 + \frac{\mu_2}{100} \times C_2 + \frac{\mu_3}{100} \times C_3 \quad (8)$$

and

$$G = \frac{C_2 - J}{C_2} \times 100 \quad (9)$$

Consider a 150 MB CAS image object with E_T = 17 MB. Assume that the user wants to secure the image using 3 security regions according to the following specifications: 4 MB are encrypted using the low encryption level (μ_1 = 2.67 %), 8 MB are encrypted using the medium encryption level (μ_2 = 5.33 %), and 5 MB encrypted according to the high encryption level (μ_3 = 3.33 %). From equation 1, we get ***PDE* = 95.33 %.**

To calculate the performance gain *G*, let C_1 = 50 performance units, C_2 = 90 performance units and C_3 = 175 performance units. From equation 8, we get $J = 11.96 \Rightarrow$ ***G* = 86.71 %.**

6. PLEDGE IMPLEMENTATION

In this section we present briefly PLEDGE's sample implementation in a real CAS environment using Gen 3 and Gen 4 EMC Centera [4] storage devices configured to operate in the Compliance Edition Plus (CE+) security mode. Building the security architecture on top of the EMC CAS storage model made it inherently comply with the HIPPA and SEC Rule 17a-4 [6] compliance regulations. The compliance requirements realized are summarized in the following points:

1. It should be impossible to change the value or transform the format of stored data.

2. It should be impossible to delete any stored object until a certain predefined retention period has expired.

3. Data shredding should be enforced to ensure that any deleted object can't be restored. EMC Centera supports data shredding by overwriting the deleted object with seven passes of random binary strings. This is also the number of passes recommended by the DoD 5015.2 regulations.

4. There should be an implementation of an event-based retention mechanism which ensures that a certain stored object can not be deleted until after a specified or predictable event takes place. This feature is very crucial in medical as well as financial, mortgage, and insurance industry operations. Event-based retention is also recommended by DoD 5015.2.

Along with all the above features implemented, PLEDGE adds the selective cryptographic protection mechanisms to secure data in transit and in the CAS storage nodes.

As mentioned previously, the application simulated is an X-Ray radiography scanning system for the secure generation, storage and retrieval of X-Ray images in a healthcare enterprise network environment. The client devices used are a) a Pentium (R) M laptop with a 2.13 GHz processor and 512 MB RAM running Microsoft Windows XP SP2, b) an Asus MyPal A730 Pocket PC device having an Intel 512 MHz processor and 64 MB RAM running Microsoft Windows Mobile for Pocket PC 2003, c) a Nokia E61i smart phone having an ARM9-based 222 MHz processor and 60 MB RAM running the Symbian 9.1 operating system. The Asus MyPal and the Nokia E61i devices come with integrated WiFi connectivity which was useful to test the interactions with the application server over the wireless network. The application server used is an Intel Pentium 4 server with a 2 GHz processor and 2 GB RAM. The application server operating system is Windows Server 2003 Enterprise Edition SP1. The protocol implementation for clients a and b used the Gen 4 EMC Centera running the CentraStar 3.1 operating code and the client c device was tested with the Gen 3 EMC Centera running the CentraStar 3.0 operating code.

Over 300 radiography digital scans were used in the sample tests ranging in size from 10 MB for standard X-Ray images to over 1.5 GB for oncology tests and studies. We utilized the expertise of many radiographers and oncologists to select reasonable regions in the X-Ray scans to secure based on the policy-driven approach proposed. The benchmarks performed are done in two main stages to calculate the overall performance of the system in terms of the number of bytes processed per second. The first stage computes the number of bytes processed per second from the time the X-Ray image is provided to the Security Engine on the client device or on the application server (depending on the configuration) to the time it is safely stored in the CAS device. The second stage calculates the number of bytes processed per second from the time the X-Ray image is queried and fetched from the CAS storage node to the time it is decrypted by the Security Engine on the client-side. The experimental results of configuration 1 are presented in Figure 6. In the secure storage stage, PLEDGE policy-based security approach achieves a speedup factor of 3 over the raw encryption approach that encrypts the entire CAS object. Almost the same speedup factor is achieved when securely retrieving the object from the CAS nodes and decrypting it on the client-side. The experimental results of configurations 2 and 3 are presented in Figures 7 and 8 respectively. The significance of these results resides in the presence of a mobile device on the client tier. The content-based security mechanisms take into consideration the limited memory, processing, and energy resources of these devices and provide the suitable encryption services based on the application requirements and data sensitivity. In configuration 2, PLEDGE realizes a performance speedup factor of 3.2 over raw encryption in the secure storage phase, and a speedup factor of 3 in the secure retrieval stage. In configuration 3, which uses a relatively low-end mobile phone, the content-based security approach attains speedup factor of 3.57 in the secure storage and 3.72 in the secure retrieval phase.

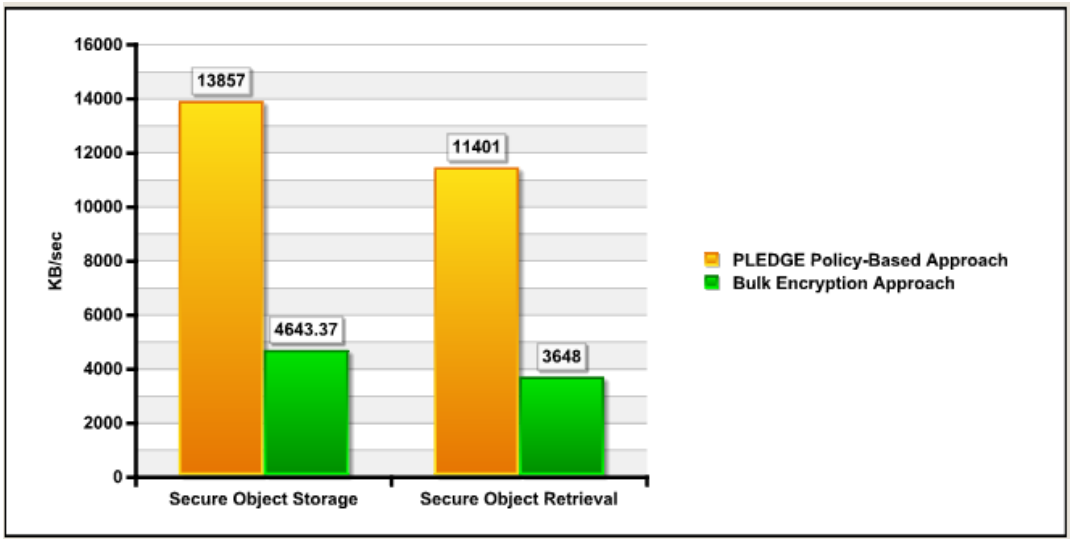

Figure 6. PLEDGE Secure Storage/Retrieval Performance
Client: Windows XP SP2 Pentium (R) M 2.13 GHz 512 MB DDR2 RAM
Application Server: Windows Server 2003 Enterprise Edition SP1 Pentium IV 2 GHz 2 GB RAM
CAS Device: EMC Centera Gen 4 Compliance Edition Plus (CE+)

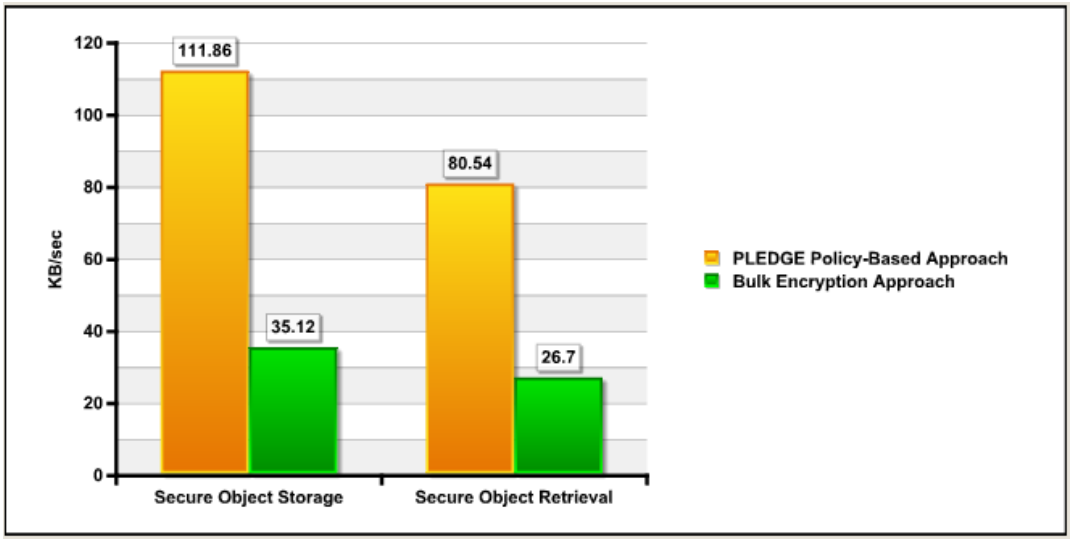

Figure 7. PLEDGE Secure Storage/Retrieval Performance
Client: Pocket PC with Microsoft Windows Mobile 2003 Intel 512 MHz Processor 64 MB RAM
Application Server: Windows Server 2003 Enterprise Edition SP1 Pentium IV 2 GHz 2 GB RAM
CAS Device: EMC Centera Gen 4 Compliance Edition Plus (CE+)

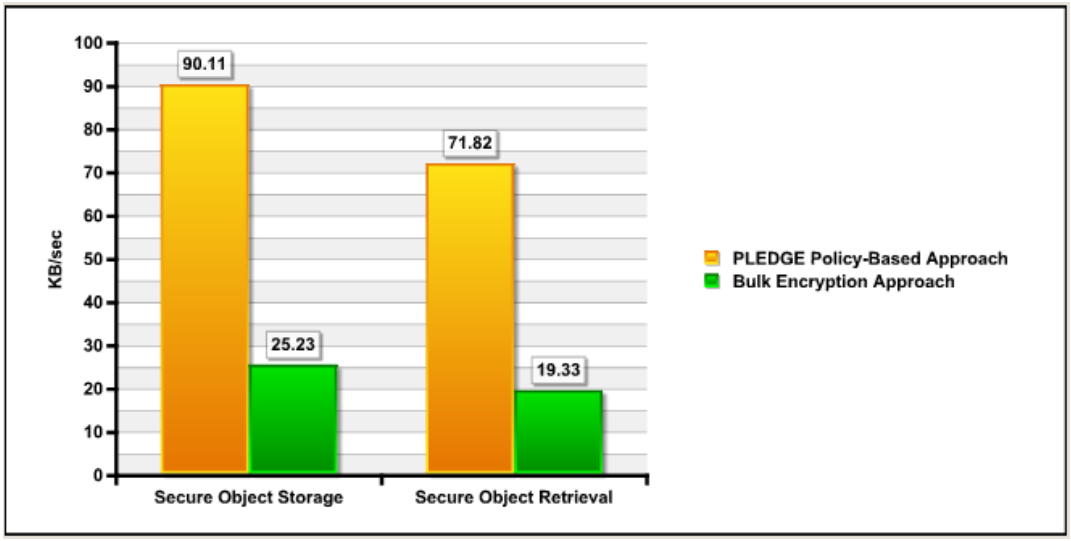

Figure 8. PLEDGE Secure Storage/Retrieval Performance
Client: Nokia E61i Smart Phone with Symbian 9.1 OS ARM 222 MHz Processor 60 MB RAM
Application Server: Windows Server 2003 Enterprise Edition SP1 Pentium IV 2 GHz 2 GB RAM
CAS Device: EMC Centera Gen3 Compliance Edition Plus (CE+)

7. CONCLUSIONS

In this paper we presented PLEDGE, a new customizable security protocol for protecting the confidentiality, integrity, and authenticity of fixed content objects in enterprise CAS environments. PLEDGE enhances the overall performance of the CAS system by utilizing a policy-driven security protocol that encrypts the CAS entity based on its content and sensitivity rather that encrypting its entire contents. PLEDGE's design and architecture are presented and a formal mathematical performance model was introduced. The paper presented a sample implementation of the protocol on a real CAS enterprise network using an EMC Centera CAS box. Implementation showed PLEDGE to be more than three times faster than bulk encryption protocols. Future research will focus on enhancing and simplifying the security policy creation mechanisms and on providing automated validation to the policy semantics to prevent any security leaks and flaws.

REFERENCES

[1] W. Itani, A. Kayssi, A. Chehab, "An efficient and scalable Security ProtocoL for protecting fixed-Content Objects in ContEnt aDdressable StoraGe architectures", *in Proc. of the Third International Conference on Security and Privacy in Communication Networks*, Nice, France, Sept, 2007.

[2] A. Freier, P. Karlton, P. Kocher, "The SSL Protocol Version 3.0, " *Internet-Draft*, 1996.

[3] T. Dierks, C. Allen, "The TLS Protocol – Version 1.0, " *RFC 2246*, 1999.

[4] EMC Centera homepage: *http://www.emc.com/products/family/emc-centera-family.htm*, accessed Jan. 14, 2008.

[5] Health Insurance Portability & Accountability Act homepage: http://www.hipaa.org, accessed Jan. 14, 2008.

[6] SEC 17 CFR Part 240, Release No. 34-38245, "Reporting Requirements for Broker Dealers Under the Security Exchange Act of 1934," January 1997,

http://www.sec.gov/rules/final/34-38245.txt, accessed Jan. 14, 2008.

[7] J. Daemen and V. Rijmen, "Rijndael, the advanced encryption standard," *Dr. Dobb's Journal*, vol. 26, no. 3, March 2001, pp. 137 - 139.

[8] National Institute of Standards and Technology, August 2002, Secure Hash Standard, Federal Information Processing Standards, Publication 180-2, http://csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenotice.pdf, accessed Jan. 14, 2008.

[9] Y. Zheng, J. Pieprzyk, J. Seberry, "HAVAL--A One-Way Hashing Algorithm with Variable Length of Output," *in Proc. Workshop on the Theory and Application of Cryptographic Techniques: Advances in Cryptology*, pp. 83-104, 1992.

[10] R. Rivest, "The MD5 Message-Digest Algorithm," RFC 1321, 1992.

[11] W. Itani, A. Kayssi, "J2ME End-to-End Security for M-Commerce," *in Proc IEEE Wireless Communications and Networking Conference*, 2003.

[12] W. Itani, A. Kayssi, "SPECSA: a Scalable, Policy-driven, Extensible, and Customizable Security Architecture for Wireless Enterprise Applications," *Computer Communications*, vol. 27, no. 18, December 2004, pp. 1825 - 1839.

[13] W. Itani, A. Kayssi, A. Chehab, "PATRIOT – a Policy-Based, Multi-level Security Protocol for Safekeeping Audit Logs on Wireless Devices," *in Proc. IEEE/CreateNet First International Conference on Security and Privacy for Emerging Areas in Communication Networks (SecureComm)*, Athens, Greece, 2005.

[14] W. Itani, A. Kayssi, A. Chehab, "An Enterprise Policy-Based Security Protocol for Protecting Relational Database Network Objects," *in Proc. 2006 International Wireless Communications and Mobile Computing Conference (IWCMC)*, Vancouver, Canada, 2006.

[15] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, F. Yergeau (September 2006), "Extensible Markup Language (XML) 1.0", World Wide Web Consortium, http://www.w3.org/TR/2006/REC-xml-20060816/, accessed Jan. 14, 2008.

[16] The Gramm-Leach-Bliley Website: http://banking.senate.gov/conf/, accessed Jan. 14, 2008.

[17] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, C. Wells and B. Zhao, "OceanStore: an Architecture for Global-Scale Persistent Storage," *in Proc. Ninth International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 190 - 201, November 2000, Cambridge, Massachusetts, United States.

[18] H. Cheng, X. Li, "Partial Encryption of Compressed Images and Videos," *IEEE Transactions on Signal Processing*, vol. 48, no. 8, pp. 2439-2451, August 2000.

[19] M. Van Droogenbroeck, R. Benedett, "Techniques for a Selective Encryption of Uncompressed and Compressed Images," *in Proc. Advanced Concepts for Intelligent Vision Systems*, pp. 90 - 97, Ghent University, Belgium, September 2002.

[20] R. Pfarrhofer and A. Uhl, "Selective Image Encryption using JBIG," *Lecture Notes in Computer Science*, pp. 98-107, 2005.

[21] S. Lian, J. Sun, D. Zhang, Z. Wang, "A Selective Image Encryption Scheme Based on JPEG2000 Codec," *Lecture Notes in Computer Science*, vol. 3332, pp. 65 - 72, 2004.

[22] X. Lu and A. Eskicioglu, "Selective Encryption of Multimedia Content in Distribution Networks: Challenges and New Directions," *in Proc. IASTED International Conference on Communications, Internet and Information Technology (CIIT 2003)*, Scottsdale, AZ, USA, November 2003.

[23] A. Pommer and A. Uhl, "Application Scenarios for Selective Encryption of Visual Data," *in Proc. Multimedia and Security Workshop*, ACM Multimedia, pp. 71 - 74, Juan-les-Pins, France, December 2002.

[24] A. Pommer and A. Uhl, "Selective Encryption of Wavelet-Packet Encoded Image Data Efficiency and Security," *ACM Multimedia Systems*, Special issue on Multimedia Security, pp. 279 - 287, 2003.

[25] M. Van Droogenbroeck, "Partial Encryption of Images for Real-Time Applications," http://www.ulg.ac.be/telecom/publi/publications/mvd/Vandroogenbroeck2004Partial.pdf, 2004, accessed Jan. 14, 2008.

[26] S. Lakshmanan, M. Ahamad, and H. Venkateswaran, Responsive Security for Stored Data, *IEEE Transactions on Parallel and Distributed Systems*, vol. 14, no. 9, September 2003.

[27] L. Wang, Y. Nie, W. Nie, and L. Jiao, "Artificial Immune Strategies Improve the Security of Data Storage", *in Proc. ICNC 2005*, LNCS 3611, pp. 839 – 848, 2005.

[28] S. Morgan, L. Russell and B. Reed, Security Method and System for Persistent Storage and Communications on Computer Network Systems and Computer Network Systems Employing the Same, International Business Machines Corporation, Patent number: 6816970, Nov 9, 2004.

[29] B. Iyer, S. Mehrotra, E. Mykletun, G. Tsudik, and Y. Wu, "A Framework for Efficient Storage Security in RDBMS," *in Proc. Seventh Int'l Conf. Extending Database Technology (EDBT 2004)*, Mar. 2004

[30] J. D. Strunk, G. R. Goodson, M. L. Scheinholtz, C. A. N. Soules, and G. R. Ganger, Self-Securing Storage: Protecting Data in Compromised Systems, *in Proc. 2000 Symposium on Operating Systems Design and Implementation (OSDI)*, October 2000.

[31] W. Diffie, P.C. van Oorschot, and M.J. Wiener, "Authentication and authenticated key exchanges", Designs, Codes and Cryptography 2 (1992), 107-125.