Context-based Encryption Applied to Data Leakage Prevention Solutions

Pilar Holgado[1], Alberto García[1], Jose Javier García[2], Jorge Roncero[2], Víctor A. Villagrá[1] and Helena Jalain[1]

[1]*Departamento de Ingeniería y Sistemas Telemáticos, Universidad Politécnica de Madrid, Avenida Complutense, 30, 28040, Madrid, Spain*

[2]*Nokia, Departamento de Innovación, Calle de María Tubau, 9, 28050, Madrid, Spain*

Keywords: Data Leakage Prevention, Context-based Encryption.

Abstract: Data leakage pose a serious threat to companies as the number of leakage incidents and the cost continues to increase. Data Leakage Prevention (DLP) has been studied to solve this information leakage. We propose a DLP solution applying context-based encryption concept, thus sensitive files are encrypted at all time. The cipher key is obtained through the execution of challenges based in the environment context and the company policies. In this paper, we explain the architecture and the design of our DLP system and the proposed challenges.

1 INTRODUCTION

Nowadays, many companies deal with sensitive data including intellectual property, financial information or users information. Accidental or unintentional distribution of private data to an unauthorized entity is a serious issue for companies. The potential damage of a data leakage can include reputation, exposure of intellectual property to competitors, or loss of future sales.

In the data leakage context, the attacker may be an internal employee or an external attacker attempting to leak sensitive information. It can be caused not only by malicious intent, but by an inadvertent mistake. In addition, an authorized user is not the same as a trusted user. In many cases organizations are victims of their own employees who intentionally share confidential data with external persons for personal purposes (Abbadi and Alawneh, 2008). In this case, the user is authorized to access to sensitive information and it is no detected from classic external measures such as firewalls.

Data Leakage Prevention (DLP) (Raman, Kayacik and Somayaji, 2011) aim to keep confidential information secure, preventing potential data leakage or unauthorized information disclosure. These solutions can be characterized according to a taxonomy that incorporates the following attributes (Shabtai, Elovici and Rokach, 2012): data-state, deployment scheme, leakage handling approach and action taken upon leakage. However, data leakage and data misuse are considered an emerging security threats to organizations, especially when carried out by insiders. In many cases, it is very difficult to detect insiders because they misuse their credentials to perform an attack.

In this paper, we propose a DLP solution based on an encryption/decryption process of confidential documents where the cipher key is obtained through the execution of *challenges*. These *challenges* use the environment context and the company policies in the encryption/decryption time. In this way, sensitive files are encrypted at all time and can only be read within our DLP system.

The rest of this paper is organised as follows. Section 2 outlines the current state of art in context-based encryption. Section 3 explains our DLP system using context-based encryption. Proposed *challenges* are explained in Section 4. Section 5 describes the key generation from *challenges* results. Finally, final remarks obtained during this study are included in Section 6.

García, A., Moro, A., García, J., Roncero, J., Villagrá, V. and Jalain, H.
Context-based Encryption Applied to Data Leakage Prevention Solutions.
DOI: 10.5220/0006475205660571
In *Proceedings of the 14th International Joint Conference on e-Business and Telecommunications (ICETE 2017) - Volume 4: SECRYPT*, pages 566-571
ISBN: 978-989-758-259-2

2 CONTEXT-BASED ENCRYPTION

Some researchers have proposed attribute based encryption (ABE) method and access controls for data privacy. ABE can be divided by two types, called KP-ABE (Key Policy Attribute Based Encryption) and CP-ABE (Cipher text Policy Attribute Based Encryption).

On the one hand, in KP-ABE each data has attributes (such as Name, Position or Place) and the users have keys based on an access tree that can distinguish attributes. For example, in (Goyal *et al.*, 2006) each user's key is associated with a tree-access structure where the leaves are associated with attributes for encryption with fine-grained access control in applications such as sharing audit log information.

On the other hand, CP-ABE can control the data access from an access tree included in each ciphertext. The methodology proposes in (Waters, 2011) allows any encryptor to specify access control in terms of any access formula (equivalently tree structures) over the attributes in the system. This access formula can be expressed in term of a Linear Secret Sharing Scheme (LSSS) and the access control included several LSSS in a matrix. However, LSSS matrices are much less intuitive to use when compared with other approaches such as boolean formulas or access trees. To address this problem, in (Liu, Cao and Wong, 2010), it is proposed a new algorithm which, in addition to AND and OR gates, can directly support threshold gates, and obtain much smaller LSSS matrices

Similarity, the user attributes have to satisfy the boolean formulas or access tree conditions in both KP-ABE and CP-ABE methods. Our proposed encryption architecture is not based on any access structure for data protection but we control data access through several complex *challenges* that return correct *sub-keys* related to environment context.

Jungyub Lee et. al. introduce the context-based encryption term in IOT (Internet of Things) scenarios (Lee, Oh and Jang, 2015). They claim that contexts related with user or device can be used as an attribute of data when data was encrypted in an ABE schema. The contexts are extracted by the detection method based on the user's situations. In the same way, we use context-based encryption for data privacy but applied to a DLP architecture.

J. Al-Muhtadi et. al. propose context and location-aware encryption for pervasive computer environment (Al-Muhtadi *et al.*, 2006). Specifically, they use a node's location to authorize access to a resource. Furthermore, the material is stored in an encrypted fashion, and can be aggregated and decrypted only when the requestor entity is at the correct location or under the correct context. The administrator sets up the spatial region(s) in which data access is authorized. Each region pre-calculates and storage a key based on the location context. This method need use certificates to establish user sessions for the authorize access process in the server. Furthermore, the data is sent between Location Service and the file system of the client. In contrast, our proposal does not need asymmetric cryptography or sessions management because it is very difficult that the attacker knows the correct context parameters for a specific file. Also, the data always remain in our encryption file system and is not transmitted.

In our proposal we apply this concept of users context, such as their location to make these parameters part of the key that will give access to an encrypted file. We execute several context *challenges* in the encryption/decryption time without storing any key or *sub-key* in a persistent form. Thus, if that specific context is not met, the generated key will be incorrect and the file could not be decrypted.

3 CONTEXT-BASED ENCRYPTION APPLIED TO DLP

The design of our DLP system is based on context data for the encryption/decryption of sensitive data. The proposal pretends to use, not only the traditional user's information such as his username or his employment, but also include the device context as the current date, GPS location, and so on. The file encryption/decryption is possible if the context has not changed when DLP system performs both processes, for example, users are working within the same hour interval, in the same place, etc. Thus, the cipher/decipher key is context-based based on the execution of multiple pieces of code called *challenges*. Each of these *challenges* generates a *sub-key* based on a specific context to obtain the final cipher key.

This kind of context-based encryption is appropriate for a DLP tool since a user, who can be authorized but not trusted, will only have access to the data when the context corresponds to the policy

of the company. Otherwise, the encipher/decipher key will be wrong.

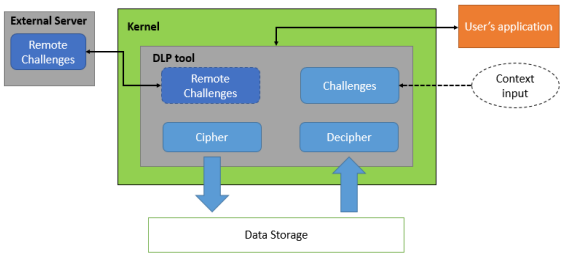

Figure 1: DLP tool architecture with server.

The architecture (Figure 1) is composed of a DLP tool and a External Server, which executes *remote challenges* and performs user authorization process.

The calculation of the challenges-based *sub-keys* in external server make the system more secure since a potential attacker trying to break the data should manipulate also that external server. Furthermore, executing *remote challenges* help to save some battery if the DLP tool is running in a mobile device. The server is an HTTP server implemented by an API-REST. Furthermore, the database stores context policies established by an administrator.

The DLP tool will generate the cipher/decipher key with every single *sub-key* returned by the *local challenges* and the *remote challenges.*

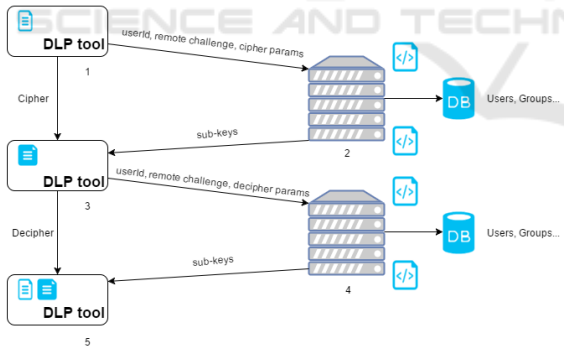

Figure 2: Flow between DLP tool and the server.

In Figure 2, we can see this behaviour in detail. The DLP tool sends a request to the server with some context information of the user for the *remote challenge* that wants to execute. The *External Server* receives this request and executes the desired *remote challenge*, accessing the database to obtain the parameters established by an administrator. Finally, the server sends back the *sub-key(s)* calculated in JSON format.

4 CHALLENGES

Challenges are different mechanisms to calculate *sub-keys* needed in our symmetric encryption process. Each *challenge* is related to a particular environment context parameter. It should be noted that the *remote challenges* in both the encryption and the decryption processes are the same, so it is not necessary to make different requests to the server. In the following subsections the proposed *remote challenges* are explained in more detail.

4.1 GPS Location Challenge

In this case, the *challenge* limits access to confidential files in all places that are not bounded within a geolocation area delimited by the system administrator and included in the database using two parameters:

- Geographical coordinates of the centre of the allowed area. For example, the centre of the office building.
- The area to be covered is circular, so the server only storage the radius of the circle to cover per system.

When a device needs to encrypt or decrypt a file it makes a POST request to the server with the geographical coordinates in which it is. First, the *challenge* using the radius and the centre stored by the manager, calculates the four points of the circle whose latitudes are higher and lower (that is, the minimum and maximum values of latitude that can be given in all points of the interior of the circle) and checks which is the common part between the points. For example, the latitude points 3.4567º and 3.4589º have in common the 3.45º part. In other words, they have in common the first 3 digits of the coordinate. This process would be done for both latitude and longitude, obtaining the number of invariant digits in each of them, generating two variables that contain the number of invariant digits in each of the dimensions of the area. Whenever you access from within this area, you get the same *sub-key* for a particular file, while outside it, a random and invalid *sub-key* value is returned.

4.2 Date Challenge

Date *challenge* limits data access according to specific date range, such as being able to decrypt files whether the date lie within the framework of the project definition. This *challenge* could be solved locally, but changing the date in a device is very easy.

Our design is based on a mask that marks the valid month range established by the administrator in the database, as is done in IP addressing. If we perform an operation of type *Month* AND *Mask* using a given *Mask*, then we always obtain the same key when the *Month* is in the correct range. To do this, the fortnights of each month will be binary coded according to their order, so that the months close to each other share the largest number of possible bits to be able to use the mask and that different date ranges can be implemented. Therefore, to encode the 24 fortnights in a year, we must use 5 bits (at least). A possible encoding would be (for fortnights): January (00000, 00001), February (00010, 00011)...

Once the fortnights of each month have been coded, we can assign different masks depending on the period of validity of the files. For example, the *Mask* 11111 is represented a period of a fortnight from the creation date of the file and the *Mask* 11100 correspondig to 2 months. Furthermore, we take into account the *day* and the *year* of creation date of each file. Thus, the key is calculated using the current *year* and a offset as the *day* value in the encryption/decryption time.

When the client encrypt/decrypt a file, the POST request to the server must include the creation date of the file as parameter to fixed the range. Then, the *challenge* obtains the *sub-key* based on current date.

4.3 Time Challenge

Time *challen*ge is checking the moment to encrypt/decrypt a file. In this way, we can limit access according to the time, such as files access limited to working hours. This *challenge* could be solved locally, but changing the time in a device is usually very easy.

The administrator includes a strip of time in the database for each department. We implement this *challenge* using a mask that marks the duration of the valid range of time, as in the date *challenge*. If we perform an operation of type *Time* AND *Mask* using a given *Mask*, then we always obtain the same key when *Time* are in the correct range. For this propose, the hours of the day will be binary coded according to their order, so that the hours close to each other share the largest number of possible bits to be able to use the mask and that different time ranges can be implemented. That is, every hour have binary representation, so 24 hours require 5 bits to be able to encode all the hours.

Once the hours are been coded, we can assign different masks depending on different time periods. For example the *Mask* 11111 is represented a period of one hour and the *Mask* 11000 is corresponding to 8 hours.

When the client encrypt/decrypt a file, the server must include the creation time of the file as parameter to fixed the range.

4.4 Wi-Fi Challenge

Wifi networks that are within reach of the equipment can be used to determine the location of the user. The administrator stores the SSID, the channel and the minimum power of the Wifi networks configured to solve the challenge in the database. Thus, we can determine where the confidential files can be accessed. The minimum power value is used to verify that the user is in the specified place, such as the company building and not on the street at a close distance.

Once Wifi networks are configured, the device makes a POST request to the server and sends the wifi networks within reach. It should be noted that the device does not know which are the good networks (to pass the *challenge*), so it has to send all wifi networks within reach, and it is the server which must verify that all the necessary are among those sent by the device. With each Wi-Fi network whose existence has been proven, a *sub-key* chunk of this *challenge* will be generated. In this way, if all the Wi-Fi networks are found, the key will be generated completely, while if any missing the generated key will be incomplete and, therefore, will not be valid to decrypt the file of the device.

4.5 Operator Challenge

If you have a list of telephone operators by country, you can check the operator of the equipment to find out which country you are in and thus have another location parameter. Typically, companies have their mobile phone service with the same company, so the operator will always be the same and will be a condition to be able to decrypt the file.

Thus, we configure the *challenge* to generate a key, doing a series of operations with the name of the operator. That is, with each operator a different key will be getting.

4.6 Robustness

Once we have all the key *challenges* calculated, together they form a complete key which is the one for encryption/decryption, since each of them by themselves are useless. To do this, the client device

needs an encryption utility or module that will calculate the final key and encrypt/decrypt the file. Thus, if an attacker would know a *sub-key* value or all *sub-keys* value, the decryption key would not be obtained.

Another case is where the attacker has a client device with our DLP system installed, either by an insider or external attacker. The main difficulty of breaking this type of system is that to calculate the correct key of a file it is necessary to fulfill all the *challenges*, since if one is not verified the corresponding *sub-key* will be incorrect and, therefore, the final key as well, i.e. the attacker have to know the correct values of all context parameter. On this way, we can avoid that attackers getting authorization or fake any context parameter obtain confidential information.

5 KEY GENERATION

The encryption module generates a key for each file in the client device. The final key must be calculated from the *sub-keys* obtained from the execution of the remote and local *challenges*, respecting two indispensable properties: computational efficiency and collision-resistance. A cryptographic hash function is a mathematical function that satisfies these two properties as well as other interesting ones including generation a fixed size output for any input size, computacional efficiency of O(n), where n is the number of bits in the string, unidirectional, and generation the same output any time it is called with a same input.

Specifically, the encryption module receives as inputs the N *sub-keys* and these are concatenated by obtaining a single 256 * N bit string. Finally, the encryption 256-bit key is generated making the SHA-256 hash of the string. In this way, files that meet the same parameters for the *challenges* would get the same *sub-keys* and therefore, the same final key. Furthermore, the algorithm will take into account a different random value for each file.

6 CONCLUSIONS

Nowadays, data leakage and data misuse are considered an emerging security threats to organizations, especially when carried out by insiders.

In addition, DLP tools in the market have usually been focused on preventing data leakage from external attackers and treats their users as trusted ones. The application of context-based encryption into a DLP tool is a big step forward to solve this problem.

Our proposal is based on the execution of several *challenges* to obtain the encryption key related to different environment context parameters. Thus, only if the authorized users comply with the context values set by the administrator, the confidential files are decrypted. This process is carried out in a transparent way to the user, who is not aware of the environment context valid for each file.

The definition of more *challenges* and the robustness study of the passwords will be included as future work.

ACKNOWLEDGEMENTS

This work has been partially funded with support from the Spanish MINECO (project DroneFS), with code RTC-2015-4064-8 and the Spanish MINETUR (project CiberNoid) with code TSI-100200-2015-035.

REFERENCES

Abbadi, I. M. and Alawneh, M. (2008) ‘Preventing insider information leakage for enterprises’, in *Emerging Security Information, Systems and Technologies, 2008. SECURWARE’08. Second International Conference on*, pp. 99–106.

Al-Muhtadi, J., Hill, R., Campbell, R. and Mickunas, M. D. (2006) ‘Context and location-aware encryption for pervasive computing environments’, in *Pervasive Computing and Communications Workshops, 2006. PerCom Workshops 2006. Fourth Annual IEEE International Conference on*, p. 6--pp.

Goyal, V., Pandey, O., Sahai, A. and Waters, B. (2006) ‘Attribute-based encryption for fine-grained access control of encrypted data’, in *Proceedings of the 13th ACM conference on Computer and communications security*, pp. 89–98.

Lee, J., Oh, S. and Jang, J. W. (2015) ‘A Work in Progress: Context based encryption scheme for Internet of Things’, *Procedia Computer Science*. Elsevier, 56, pp. 271–275.

Liu, Z., Cao, Z. and Wong, D. S. (2010) *Efficient generation of linear secret sharing scheme matrices from threshold access trees*.

Raman, P., Kayacik, H. G. and Somayaji, A. (2011) ‘Understanding data leak prevention’, in *6th Annual Symposium on Information Assurance (ASIA 11)*, p. 27.

Shabtai, A., Elovici, Y. and Rokach, L. (2012) *A survey of data leakage detection and prevention solutions*. Springer Science & Business Media.

Waters, B. (2011) 'Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization', in *International Workshop on Public Key Cryptography*, pp. 53–70.