Available online at www.sciencedirect.com

ScienceDirect

IFAC-PapersOnLine 49-25 (2016) 241–246

Client side data encryption/decryption for web application

Václav Kaczmarczyk*. Zdeněk Bradáč**.
Petr Fiedler***. Jakub Arm****

*Faculty of electrical Engineering and Communication, Brno University of Technology, Brno, Czech Republic
(e-mail: kaczmarczyk@feec.vutbr.cz)
**Faculty of electrical Engineering and Communication, Brno University of Technology, Brno, Czech Republic
(e-mail: bradac@feec.vutbr.cz)
***Faculty of electrical Engineering and Communication, Brno University of Technology, Brno, Czech Republic
(e-mail: fiedlerp@feec.vutbr.cz)
****Faculty of electrical Engineering and Communication, Brno University of Technology, Brno, Czech Republic
(e-mail: xarmja00@stud.feec.vutbr.cz)

Abstract: This paper describes an approach and implementation issues of client-side data encryption for thin client – web browser. Since the service provider may be a competitor of the service user, sensitive parts of data are encrypted and decrypted by the client. Cipher key generation and distribution issues are discussed: although the cipher key is kept in private, the paper shows the way for the key validation by the server side to maintain data consistency.

Keywords: **Client-side encryption, Thin client, AES, SHA256, jQuery, Key validation**

1. INTRODUCTION

The problem of data classification and security is one of the most important problems that need to be solved today in the field of information systems. Protection of sensitive information constitutes one of the most crucial tasks for systems used in public health service, civil services and many other areas. Although sensitive data need to be safely stored and protected, they must remain accessible for authorized reading and modification. To overcome this challenge, encrypted data transfer, key management and other approaches must be employed.

This paper describes information system that employs not only traditional data protection by secured data transfer (SSL, TLS protocols) and login-password authorization. Moreover the part of data stored within system database can be encrypted and protected from abuse even by the service provider (database administrator). The example shown in this paper involves an elevators database service shared through multiple enterprises. Within this database there are both sensitive (personal data) and technical data for each elevator. Whereas sensitive data must be encrypted in the database (e.g. some identifier, elevator installation address, owner information), technical data encryption is not requested. The service provider owns the full access to the database. Other clients can access to data through the online service exclusively. The sensitive data encryption is required due to possible enterprises competing, since the service provider and service clients are in the same branch. Since elevator technical data remain unencrypted, the service provider (which may be elevators manufacturer) can provide a technical support for their elevators. Moreover using anonymized set of data allows service provider to improve the service quality and even to improve their products.

1.1 Terminology

Within this section we clarify terminology used within the rest of this paper. Server means here the application running on the ASP.NET server and offering the service to clients. All data are stored in MySQL database. Client means the web browser supporting HTML5 and JavaScript. Reasons that led to use the thin client approach are mentioned in the table 1. Server application is based on the ASP.NET Web Forms technology that allows dynamic web pages generation. This technology ensures web pages compatibility with wide range of browsers and devices since the web page is customized based on the specific browser demands. ASP.NET technology is based on Microsoft .NET Framework (Mikolajek, M., Otevrel, V., Koziorek, J., and Slanina, Z. 2015). Hence it can use all advantages the framework provides. However, the target system for running server application must be the Microsoft Windows (Server) operating system (Spaanjaars, 2010). Alternativelly, the whole solution can be deployed to Microsft Azure platform and run remotely (Microsoft, 2016).

Service provider in the following is an enterprise that provides the server hardware, software and database. Service clients are enterprises that perform elevators service for end customers (enterprises employee). Moreover, the service can be used also by enterprise customers to access information (statistics, faults, service and maintenance details) of their own elevator(s). The service can provide a communication channel between end users and their service enterprise.

10.1016/j.ifacol.2016.12.041

Table 1. Thick and thin client comparation

Thin client	Thick client
(+) Always current client version compatible with server counterpart	(+) Client application under full control, unbounded implementation possibilities
(+) OS and Platform independency	(-) Version dependency – necessary to distribute/update current client version.
(+) Wide range of UI design ability (extension by JS libraries, HTML5 technology).	(-) Client OS and platform dependent
(+) Transparent solution (client code publicly accessible - JavaScript).	
(-) Must rely on standardized communication solutions (REST API)	

2. BACKGROUND

The aim of our work is to enable client-side (web browser side) data encryption, presentation, consequent decryption and update. This approach allows such processing sensitive data that only authorized person with encryption key can have access granted. Unencrypted data are never visible to the server. Moreover, the server does not process encryption key. Since data are encrypted by symmetric cipher, it is difficult to decrypt them without the key. The encryption key is always common for one client (enterprise staff and end users – enterprise customers). Moreover each person using the service owns its unique user name and password to access the application.

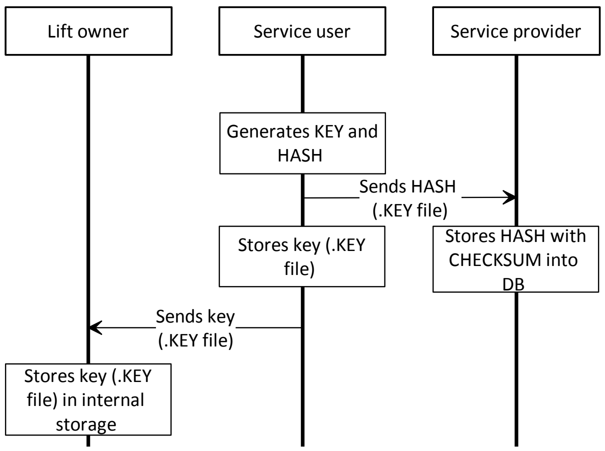
2.1 Encryption key

Since data are encrypted/decrypted by the client side solely, we decided to use symmetric ciphering for our work. Moreover, symmetric ciphering also simplifies cipher key distribution issues. After the simple research we concluded to use Advanced Encryption Standard (AES, so called Rijndael). For AES the encryption key of 128, 192 respective 256 bits is needed. For the security reasons we decided to use AES 256 algorithm and hence the key must be 256 bits long. However, such a key coded in Base64 can be represented by approximately 43 alphabetical characters, which is impractical for several reasons.

The user must not be challenged to type such a long key – we tested this by the sample of 10 users and no one of them was able to type the key for the first time correctly (Moreover, they also were not happy during this test). In case that incorrect key is entered by user and this key is not validated, the system performs data decryption by the wrong key and presents meaningless information. Moreover, if data were encrypted by the wrong key and stored into the database subsequently, enterprise data integrity would be harmed. In case that user made a typo and cannot reproduce it again, data would be inevitably lost.

To overcome this problem we decided to distribute and keep the cipher key within a special file (see picture 1). The user (authorized enterprise representative) uses a special application to generate a strong random key (the key generation includes random numbers from mouse move generation, the approach described within (Barker, 2012) was used). The generated key is equipped by the control field that helps to determine its corruption and stored into the file (.key extension). Together with the cipher key the validation file is generated. This file contains the SHA-256 hash of the key (Savard, 2016) together with its own control field (.validationHash extension). As the SHA-256 is well known hash algorithm, it is quaranteed that the cipher key cannot be computed back from its result. While the key file shall be kept in secret in the scope of the enterprise and authorized users, the validation file must be delivered to service administrator. Before enterprise users can start using data encryption, the validation file must be placed into the database.

Picture 2. Key generation and distribution



Since the enterprise validation hash is stored within the database, service users (employee, lift owners) that belong to this particular enterprise can access to encrypted information. Through the web interface the user uploads the .key file (see the picture 2). Once the upload is finished, the key file is checked against the control field and if considered as correct, the key is stored into the LocalStorage object accessible through JavaScript client code (the HTML5 support is required for proper service function) (Joreteg, 2016).

The entire client-side functionality is implement as JavaScript code (interpreted by the web browser), hence its function can be easily validated by the interested service user.

2.1 Client-side data encryption and decryption

Once the key file is loaded into the web browser local storage the particular user can get access to encrypted data. The communication technique is shown in the picture 3. When the service user tries to get the page with potentially sensitive data, the following is applied.

1. Client sends a request for the specific webpage (e.g. the list of all owned lifts).