1-22-2018

DISTRIBUTED SYSTEM MODEL FOR KEY MANAGEMENT

Mersaid Aripov prof.
National University of Uzbekistan named after Mirzo Ulugbek, mirsaidaripov@mail.ru

Ruhillo Habibovich Alayev
National University of Uzbekistan named after Mirzo Ulugbek, mr.ruhi@outlook.com

Follow this and additional works at: https://uzjournals.edu.uz/tuitmct

Recommended Citation
Available at: https://uzjournals.edu.uz/tuitmct/vol1/iss1/5

This Article is brought to you for free and open access by 2030 Uzbekistan Research Online. It has been accepted for inclusion in Bulletin of TUIT: Management and Communication Technologies by an authorized editor of 2030 Uzbekistan Research Online. For more information, please contact brownman91@mail.ru.
DISTRIBUTED SYSTEM MODEL FOR KEY MANAGEMENT

Aripov M.M., Alaev R.H.

Key management plays a crucial role in cryptography, as the basis for secure information exchange, data identification and integrity. There are software and hardware key management tools that support Crypto APIs and Cryptography Next Generation APIs (CNG API), Public Key Cryptography Standards (PKCS). These tools store cryptographic keys on hard disks, smart cards, tokens, and in other storage media. To use the cryptographic keys stored on these smart cards and tokens, you need to connect them to the appropriate hardware. The cryptographic keys stored on the hard drives of a computer or a laptop are used by the programs of these devices. If it becomes necessary to use a single key in different systems, then you will have to create copies of the key on all these devices. This complicates the process of key management, raises tasks of securely store keys, keys access control.

This paper proposes a distributed system model for key management and a protocol of interaction of the distributed system modules. The proposed model provides the ability to store keys in a smartphone, and access to keys from other devices. The system described in the model consists of 3 modules. The module 1 has computer version and smartphone version, and serves to send a request for signing, signature verification, hashing. The module 2, a smartphone software, provides key pair generation, storing, encrypting and decrypting, archiving keys, export/import keys, keys access control, and destroying keys. The module 3, web service, provides communication of the first and second modules.

In addition, the system, which was created based on the current model, provides the ability to use digital signatures in web applications. The Module 1 operates as a local web service that accepts requests from a web page running in a browser. A special script in a web page sends http requests that include cryptographic operations to the specified localhost port and accepts responses.

Keywords: key access control, key management, digital signature, hashing, encryption, decryption, QR Code, smartphone, web service.

Introduction

Legal entities and individual entrepreneurs are required electronic digital signature for using most of the interactive services of state organizations in our country. Electronic digital signature is used for identify, verify the integrity of electronic documents, etc. Digital certificate and private key are provided in an encrypted PFX format. Digital signature owners store PFX file in flash media and in computer. Practical implementation of storing digital certificate and private key in tokens or in smart cards are very difficult for the majority of the population. Interactive services are offered in the form of web services, web applications and / or mobile applications. For some interactive services it is convenient to use a computer or laptop, and for others a smartphone. Thus, it becomes necessary to use the key in different devices for different types of applications. Considering that the majority owns smartphones, we offer a method of storing digital certificates and private keys in smartphones, and a model of accessing keys for signing and verification signatures from different computers, laptops and smartphones. In our model, we used a cryptographic module in a smartphone that is responsible for key management, a cryptographic module in a laptop or a computer that generates a signing request and signature verification, as well as an intermediate web service to ensure interactions of these cryptographic modules. The architecture of the system, which provides implementation of the proposed model, is shown in figure 1.

Cryptographic key management encompasses the entire lifecycle of cryptographic keys and keys materials. Basic key management guidance was provided in [1]. A novel Key-Lifecycle Management System was presented in [2]. The presented Key-Lifecycle Management System introduces a pattern-based method to simplify and to automate the deployment task for keys and certificates, also it provides a novel form of strict access control to keys and realizes the first cryptographically sound and secure access-control policy for a key-management interface. Developing a cryptographic key management system for distributed networks was discussed in [3]. A Time-based Group Key Management algorithm for cryptographic cloud storage applications, which uses the proxy re-encryption algorithm to transfer major computing task of the group key management to the cloud server was proposed in [6]. The algorithm described in [6] ensures data confidentiality when storing user data in an unreliable cloud using encryption and proxy.

Scholars and management consultants have identified platform control as a key feature for business success in the ICT industries [10].

The introduction of public key cryptography was a critical advance in IT security. It enables confidential communication between entities in open networks, in particular the Internet, without prior contact [12]. A Lightweight Public Key Infrastructure (LPKI) for the mobile phones was introduced in [7]. It
provides robust distributed authentication services. A method for distributed E-Business application authentication using public key cryptography was introduced in [8]. An enhanced secure authentication protocol for roaming services on elliptic curve cryptography was proposed in [9]. The proposed protocol’s formal security is verified using Automated Validation of Internet Security Protocols and Applications tool to certify that the proposed protocol is free from security threats. A distributed e-business authentication scheme based on conic curve was proposed in [11]. A new approach to setting up a secure authentication and authorization procedure in a distributed computing system without proxy certificates was proposed in [13]. However, these works do not include a solution for the above problem.

Figure 1. The architecture of the system

The QR code is used in this system for exchanging password between computer and smartphone. In the work [4], QR code was analyzed from the perspective of their significance and uses. In [5] Shettar found QR code as a great technology which helped library to cater most crucial user demand for access to information through mobile phones. The QR code was discussed in [14,15]. The using QR code for ensuring security such as authentication, authorization was discussed in [19-23]. The implementation of the QR code in some areas such as payment system, government service, library was provided in [16-18].

Main part

We introduce the following notation:

- \( E \) – encryption function of public key algorithm
- \( E' \) – encryption function of cipher algorithm
- \( D \) – decryption function of public key algorithm
- \( D' \) – decryption function of cipher algorithm
- \( h(x) \) – hash function
- \( M \) – data to sign
- \( H \) – hash value
- \( K_{pu} \) – user public key
- \( K_{ps} \) – user private key, private key is saved in encrypted form
- \( K_{w} \) – web service public key
- \( K_{pw} \) – web service private key
- \( K_{se} \) – symmetric keys
- \( C_{c} \) – crypto module in computer, it has built-in certificate of \( W \) service
- \( C_{m} \) – crypto module in smartphone, tablet for generation, export, import, destroying key pairs and signing hash, it has built-in certificate of \( W \) service
- \( L \) – login of user in \( W \), \( L \) maybe phone number, e-mail etc.

Encryption private key of user:

\[
K = h(h(\text{PINcode})), \ K_{pu}' = E'_{K_{pu}} (K_{pu}) , \ H_{k} = h(K).
\]

\( \{K_{pu}, K_{ou}, H_{k}\} \) - encrypted private key, public key and hash value of PIN code. Hash value is used for checking the PIN code.

Decryption private key of user:

\[
K = h(h(\text{PINcode})), \text{ if } H_{k} = h(K) \text{ then } K_{pu} = D'_{K} (K_{pu}').
\]

The following processes describe interactions between modules step by step.

Registration process

The registration process is done with modules \( C_{m} \) and \( W \) service. It includes the following steps:

Step 1: \( C_{m} \) asks from user to select generation new key pairs or import an existing one.

Step 2: \( C_{m} \) asks user to enter PIN code.

Step 3: If user selects import an existing key pair go to step 7.
Step 4: $C_m$ generates new key pairs $\{K_{pu}, K_{ms}\}$. $C_m$ encrypts private key, saves $\{K_{pu}, K_{ms}, H_k\}$ to storage.

Step 5: $C_m$ generates Certificate Signing Request (CSR) and sends CSR to Registration Authority (RA).

Step 6: $C_m$ installs certificate obtained from RA, go to step 9.

Step 7: $C_m$ asks user to enter password, and imports key pairs encrypted in PFX format.

Step 8: $C_m$ encrypts $K_{pu}$, saves $\{K'I_{pu}, K_{ms}, H_k\}$ and certificate to storage.

Step 9: $C_m$ asks user to enter $L$, calculates $H = h(L)$, $H' = E_{K_{pu}}(H)$ and sends $\{L, H', user certificate\}$ to $W$.

Step 10: $W$ checks if $D_{K_{pu}}(H') = h(L)$ go to step 11, else returns "Incorrect parameters".

Step 11: if $L$ is not used by other user, then go to step 16.

Step 12: $W$ checks “Are these certificates belongs to this user or not”. For this purpose $W$ selects all of certificates attached to $L$, generates $T$ random number, calculates $T' = E_{K_{pu}}(E_{K_{ms}}(T))$ and returns $\{T', certificates\}$ to $C_m$.

Step 13 $C_m$ checks if it has any certificate that is in the list of certificates returned by $W$, then it calculates $T = D_{K_{pu}}(D_{K_{ms}}(T'))$, $T'' = E_{K_{pu}}(E_{K_{ms}}(T))$ and sends $\{T'', selected user certificate\}$ to $W$, go to step 15, else go to step 14. In this case $K_{pu}^1$ is private key of the selected certificate by $C_m$.

Step 14: $C_m$ shows to user the message “Login is already taken”. Go to step 9.

Step 15: $W$ checks, if $T = D_{K_{pu}}(D_{K_{ms}}(T''))$.

Step 16: $W$ registers user, saves $L$ and user certificate in database, returns “ok” as status of registration, go to step 17.

Step 17: Registration is completed successfully. $C_m$ closes the connection with $W$.

**Signature process**

All 3 modules take part in the signature process. This process includes the following steps:

Step 1: $App$ opens connection with $C_i$ for signing.

Step 2: $C_i$ asks to enter $L$ from user.

Step 3: $C_i$ generates one time password $P$ and shows it to user as QR code.

Step 4: $C_m$ scans QR code and gets $P$. $C_m$ checks if user has more than one certificates, $C_m$ asks user to select one of them, else uses that one. $C_m$ asks user to enter PIN code for access to private key.

Step 5: $C_m$ asks user to enter PIN code for access to private key. $C_m$ calculates $K = h(h(PIN code))$, if $H_k = h(K)$ then $K_{pu} = D'_K(K'I_{pu})$, else shows “PIN code is incorrect” message and go to step 5.

Step 6: $C_m$ calculates $H'_m = h(P)$, $S' = E_{K_{pu}}(H'_m)$. $C_m$ generates $K_{ms}$, calculates $P'_m = E_{K_{ms}}(P)$, $K'_{ms} = E_{K_{ms}}(E_{K_{ms}}(K'_{ms}))$.

Step 7: $C_m$ sends $L$ and user selected certificate to $W$.

Step 8: $W$ generates random number $T$, calculates $T' = E_{K_{pu}}(E_{K_{ms}}(T))$, and returns $T'$ to $C_m$.

Step 9: $C_m$ calculates $T = D_{K_{pu}}(D_{K_{ms}}(T'))$, $T'' = E_{K_{pu}}(E_{K_{ms}}(T))$ and $C_m$ sends $\{L, T', P'_m, K'_{ms}, S'\}$ to $W$.

Step 10: $W$ checks if $T = D_{K_{pu}}(D_{K_{ms}}(T''))$ then calculates $K_{ms} = D_{K_{pu}}(D_{K_{ms}}(K'_{ms}))$. $P = D'_{K_{ms}}(P'_m)$, $H = D_{K_{ms}}(S')$ and checks if $H = h(P)$ than $W$ sets temporary password $P$ for $L$.

Step 11: $C_i$ generates $K_{ci}$, generates $P' = E'_{K_{ci}}(P)$, $K'_{ci} = E'_{K_{ci}}(K'_{ci})$. $C_i$ and sends $\{L, K_{ci}, P'\}$ to $W$ for authentication.

Step 12: $W$ calculates $K_{ci} = D_{K_{pu}}(K'_{ci})$. $P = D'_{K_{ci}}(P')$ and checks, if $L$ and $P$ is right $W$ returns to $C_i$ the user certificate else returns to $C_i$, “Login or pass is incorrect”.

Step 13: If $C_i$ gets message “Login or pass is incorrect” from $W$, then $C_i$ shows message to user and go to Step 2.

Step 14: $C_i$ returns to $App$ the user certificate.

Step 15: $App$ sends to $C_i$ the user certificate and $M$ for hashing.

Step 16: $C_i$ calculates $H = h(M)$ and returns it to $App$, the hash algorithm determined by the certificate for signing.

Step 17: $App$ sends $H$ to $C_i$ for sign.

Step 18: $C_i$ calculates $H'_c = E'_{K_{ci}}(H)$ and sends $H'_c$ to $W$.

Step 19: $W$ calculates $H = D'_{K_{ci}}(H'_c)$, $H'_m = E'_{K_{ms}}(H)$.

Step 20: $C_m$ gets $H'_m$ and the selected user certificate from $W$.

Step 21: $C_m$ calculates $H = D'_{K_{ms}}(H'_m)$. $C_m$ asks user to allow signing $H$.

Step 22: If user allows signing then $C_m$ calculates $S = E_{K_{pu}}(H)$, $S' = E_{K_{ms}}(S)$, sends $S'$ and “ok” as status of the signing else sends “sign not allowed” as status of the signing to $W$.

Step 23: $C_i$ gets status and $S$ from $W$.

Step 24: $C_i$ returns to $App$ the status and $S$.

Step 25: $App$ checks, if status is ok $App$ uses $S$ as signature.

Step 26: $App$ closes connection with $C_i$. 

Aripov M.M., Alayev R.H.

2018, 1 (43)
Step 27: \( C_c \) closes connection with \( W \).
Step 28: \( W \) deletes \( P \).

**Signature verification**

Signature verification process includes the following steps:

Step 1: \( App \) opens connection with \( C_c \) for verify signature.
Step 2: \( App \) sends to \( C_c \) the user certificate and \( M \) for hashing.
Step 3: \( C_c \) calculates \( H = h(M) \) and returns \( H \) to \( App \).
Step 4: \( App \) sends \( H \), \( S \) and the user certificate to \( C_c \).
Step 5: \( C_c \) checks, If \( H = D_{K_c}(S) \) then \( C_c \) returns “signature is valid” as status of verification, else returns “signature is not valid” as status of verification.
Step 6: \( App \) closes the connection with \( C_c \).

**Conclusion**

In this paper, we provided a description of the distributed system model for key management and a protocol of interaction of the distributed system modules. We presented the registration process of the user, signature process, signature verification process step by step. The user registration process includes key pair generation and/or import existing key pairs, generating and sending CSR to obtain digital certificate from RA, and registration user digital certificate in Web service. Signature process includes temporary password generation, generating QR code of the password, scanning QR code by \( C_n \), setting temporary password for session, generating and sending request for sign hash, sign hash, as well as secure data exchange between modules using encryption with cipher algorithm and public key algorithm. Signature verification process includes computing hash value and signature verification. As a digital signature algorithm [24], must be chosen one that supports encryption with public key, because key pair of the user digital certificate are used for secure key exchange, for data integrity and identification in this system. RSA and DSA, Steps presented above are enough to provide signature and signature verification using key pair in smartphone. The distributed system, presented in this paper, can be used for providing the ability to store keys in a smartphone, and accessing to keys from other devices.

**REFERENCES**


Aripov M.M., Alaev R.H.
2018, 1 (43)


Aripov M.M., Alauy R.H. 2018, 1 (43)