УДК 004.056

DISTRIBUTED SYSTEM MODEL FOR KEY MANAGEMENT

Aripov M.M., Alaev R.H.

This article proposes a distributed system model for key management. 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. 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, archiving keys, encrypting and decrypting, export/import keys, keys access control, and destroying keys. The module 3, web service, provides communication of the first and second modules.

Keywords: key access, key management, digital signature, QR Code, smartphone.

Ушбу мақолада криптографик калитларни бошқаришнинг тақсимланган тизими модели таклиф қилинган. Таклиф қилинаётган модел калитларни смартфонларда сақлаб, ундан бошқа қурилмаларда фойдаланиш имконини яратади. Моделда тавсифланган тизим тақсимланган тизим ҳисобланиб, 3 та модулдан иборат. 1-модул компьютер ва смартфонларда ишлайдиган версияларга эга бўлиб, электрон рақамли имзони яратиш учун сўровларни узатиш, электрон рақамли имзони текшириш, хэш қийматни ҳисоблаш учун хизмат қилади. 2-модул смартфонда ишлашга мўлжалланган бўлиб, асимметрик алгоритмлар жуфт калитларини генерациялаш, сақлаш, архивлаш, калитлардан шифрлаш ва дешифрлаш учун фойдаланиш, экспорт/импорт қилиш, калитга мурожаатни бошқариш, ҳамда калитларни ўчириш функцияларини тақдим этади. 3-модул веб сервис ҳисобланиб, 1- ва 2-модуллар алоқаларини таъминлаш учун хизмат қилади.

Таянч иборалар: калитга мурожаат, калитни бошқариш, электрон рақамли имзо, QR код, смартфоне.

Настоящая статья предлагает модель распределенной системы для управления ключами. Предложенный модель предоставляет возможность хранения ключей в смартфоне, и пользования ключами с других устройств. Система, описанный в модели, состоит из 3 модулей. 1-модул имеет версии, работающие в компьютерах, смартфонах, и служить для отправки запроса на подписывание, проверки подписи, вычисление хеша данных. 2-модул, функционирующий на смартфоне, предоставляет функции генерации, хранения, архивирования, шифрование и расшифрование ключами, экспорт/импорт ключей, контроль доступа к ключам, уничтожении ключей. 3-модул, веб сервис, обеспечивает связ первого и второго модуля.

Ключевые слова: доступ к ключу, управления ключами, цифровая подпись, QR код, смартфон.

I. INTRODUCTION

Legal entities and individual entrepreneurs are required electronic digital signature for using most of the interactive services of state organizations in Uzbekistan. 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.

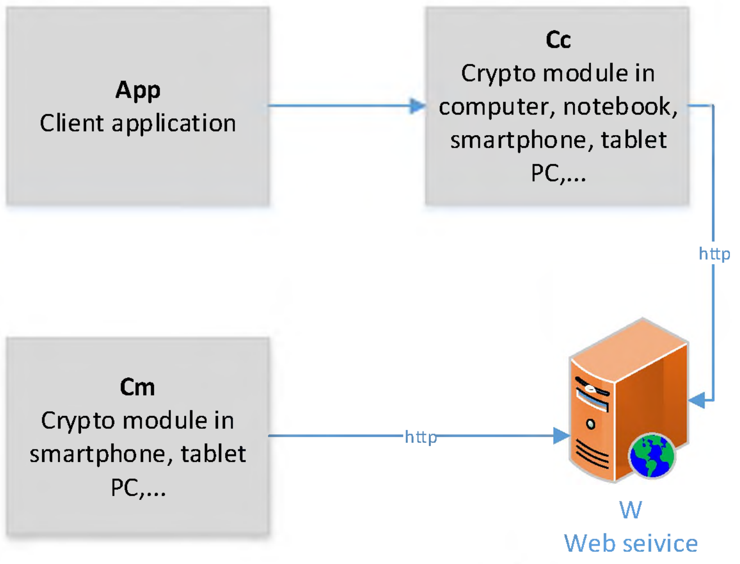
App
Client application
Cc
Crypto module in computer, notebook, smartphone, tablet PC,...
http
Cm
Crypto module in smartphone, tablet PC,...
http
W
Web seivice

Figure 1. The architecture of the system

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 (KLMS) was presented in [2]. The presented KLMS introduces a pattern-based method to simplify and to automate the deployment task for keys and certificates, also 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].

The QR code is used in this system for data reading from computer display by smartphone. In the work [4], QR codes was analyzed from the perspective of their significance and uses. In [5] Shettar found QR codes as a great technology which helped library to cater most crucial user demand for access to information through mobile phones.

II. 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_{ou} – user public key
K_{pu} – user private key, private key is saved in encrypted form
K_{ow}– web service public key
K_{pw} – web service private key
K_{ms} and K_{cs} – symmetric keys
C_c– crypto module in computer, it has built-in certificate of W service
C_m – crypto module in smartphone for generation, export, import, destroying key pairs and signing hash, it has built-in certificate of W_c service
L-login of user in W, L maybe phone number, e-mail etc.
W - Intermediate web service to provide interaction between cryptographic modules C_c and C_m. User certificates, username and temporary user password are stored here.
Encryption private key of user:

$$K = h(h(PINcode)),\ K'_{pu} = E'_K(K_{pu})\ ,\ H_k = h(K).$$

$\{K'_{pu}, K_{ou}, H_K\}$ - 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:

$K = h(h(PINcode))$, if $H_k = h(K)$ then $K_{pu} = D'_K(K'_{pu})$

Registration process

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 new key pairs generation go to step 4 else go to step 7.

Step 4: C_m generates new key pairs $\{K_{pu}, K_{ou}\}$. C_m encrypts private key, saves $\{K'_{pu}, K_{ou}, H_k\}$ 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 $\{K'_{pu}, K_{ou}, H_k\}$ 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 $D_{K_{ou}}(H') = h(L)$ go to step 11, else returns "Incorrect parameters".

Step 11: if L is not used by other user, then go to step 16 else go to step 12.

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 $T' = E_{K_{pw}}(E_{K_{ou}}(T))$, 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 $T = D_{K_{pu}}(D_{K_{ow}}(T'))$, $T'' = E_{K^1_{pu}}(E_{K_{ow}}(T))$ and sends $\{T''$, selected user certificate$\}$ to W, go to step 15, else go to step 14. In this case K^1_{pu} 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 $T = D_{K_{pw}}(D_{K^1_{ou}}(T''))$, than go to step 16.

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.

Sign process

Step 1: *App* opens connection with C_c for signing.

Step 2: C_c asks to enter L from user.

Step 3: C_c 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 $K = h(h(PINcode))$, if $H_k = h(K)$ then $K_{pu} = D'_K(K'_{pu})$, 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 $P'_m = E'_{K_{ms}}(P)$, $K'_{ms} = E_{K_{pu}}(E_{K_{ow}}(K_{ms}))$.

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

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

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

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

Step 11: C_c generates K_{cs}, calculates $P'_c = E'_{K_{cs}}(P)$, $K'_{cs} = E'_{K_{ow}}(K_{cs})$. C_c and sends $\{L, K'_{cs}, P'_c\}$ to W for authentication.

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

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

Step 14: C_c returns to App the user certificate.

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

Step 16: C_c 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_c for sign.

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

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

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_c gets status and S from W.

Step 24: C_c 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_c.

Step 27: C_c closes connection with W.

Step 28: W deletes P.

Signature verification

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*. hash algorithm determined by the user certificate.

Step 4: *App* sends H, S and the user certificate to C_c.

Step 5: C_c checks, If $H = D_{K_{ou}}(S)$ 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.

III. CONCLUSION

This article provides a description of the distributed system model for key management. The registration process of the user, signature process, signature verification process are presented in this paper. As a digital signature algorithm, should be chosen one that supports encryption with public key. 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

[1] NIST Special Publication 800-21, Guideline for Implementing Cryptography in the Federal Government, Annabelle Lee, Security Technology Group -Computer Security Division -National Institute of Standards and Technology Gaithersburg, MD 20899-8930.

[2] Björkqvist M. et al. (2010) "Design and Implementation of a Key-Lifecycle Management System", International Conference on Financial Cryptography and Data Security, pp 160-174.

[3] Acar T., Belenkiy, M., Ellison, C., & Nguyen, L. (2010). Key management in distributed systems. Microsoft Research (pp. 1–14). Retrieved from http://docplayer.net/11794546-Key-management-in-distributed-systems.html

[4] Shettar, I. M., (2016) Quick Response (QR) Codes in Libraries: Case study on the use of QR codesin the Central Library, NITK. Proc. TIFR-BOSLA National Conference on Future Librarianship-2016, 129-134.

[5] Sangeeta Singh, "QR Code Analysis", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 6, Issue 5, May 2016.