Access Control System Using Multi-factor Authentication

T. Cvrček[1], and P. Dzurenda[1]

[1]Department of Telecommunications, Brno University of Technology, Brno, Czech Republic

E-mail: tadeas.cvrcek@vutbr.cz, dzurenda@vut.cz

Abstract—A secure user authentication process is a key prerequisite for ensuring the security of the entire electronic system. On the other hand, current systems usually deploy many constrained devices with limited computational power, memory space and cryptographic support. This makes it hard to deploy secure cryptographic mechanisms in this environment. In this article, we present our multifactor authentication system using a reader with a secure module represented with MultOS smart card and an Android smart phone acting as a user authentication device. The system supports NFC (Near Field Communication) communication interface for intermediating communication between smart phone and reader, supports additional authentication factors (e.g. PIN code or fingerprint) and is easily implementable even on very constrained devices such as smart cards.

Keywords—MultOS, Android, Authentication, Secure Access Module, Multi-Factor, Access Control, Cryptography, Security, Smart Card, Authenticated Key Agreement

1. INTRODUCTION

Nowadays, the environment of Internet of Things (IoT) is growing significantly and the Industry 4.0 one is becoming more relevant. However, both environments require to be secure, to connect sensors and gadgets to common network. Losing data or compromising the network would mean possibility of losing assets, or even threat subjects themselves. On the other hand, authentication in the IoT environment is limited by constrained devices. These devices have limited performance due to their size and power consumption requirements. However, it is still necessary to provide authentication methods even in such environment.

One of the best ways to authenticate an entity is to use Multi-Factor Authentication (MFA), which is becoming important across the services we use especially nowadays. According to the directive 2015/2366, the European Union (EU) requires users to use MFA for money transactions with nominal value of 50 Euros and above [1]. MFA defines three factor types: knowledge (e.g., Personal Identification Number (PIN) and password), possession (e.g., smart card, smart phone and wearables), and inherent (e.g., behavioral and physical biometrics). According to the EU law, at least two of these types must be used.

In this paper, we developed and implemented a multi-factor access control system based on Authentication and Key Agreement (AKA) protocol published in [2]. Our implementation deploys a hardware reader with a slot for Secure Access Module (SAM) and internal clock, and the Android smart phone acting as a user authentication token. The proposed solution is secure, efficient, and easy to run on IoT devices with different computational power.

2. MULTI-FACTOR ACCESS CONTROL SYSTEM

The implemented system deploys Google Android based smart phone with Near Field Communication (NFC) support, SAM module in a form of smart card, and access terminal (i.e., single board computer or microchip embedded in the door card reader). The terminal provides communication interface between the SAM and smart phone. This terminal is connected to an electric door lock, that in case of successful authentication allows the user to open the door and access protected area. For development purposes, the Raspberry Pi 3 Model B+ has been used as the terminal. The communication protocol between the terminal and SAM/smart phone has request-response communication model and uses Application Protocol Data Units (APDU). The protocol is used for communication between a smart card and a smart phone, because both have required software support and they can not communicate directly.

The system uses SAM module. This module is integrated in the terminal and provides hardware acceleration of cryptographic primitives and secure memory management. As SAM module, one can use many available smart card platforms. In our case, we use MultOS smart card. The authentication protocol is based on AKA protocol [2] and allows us to perform two factor authentication, namely: 1) possession (i.e., smart phone) and knowledge (i.e., PIN code) or 2) possession (i.e., smart phone) and biometric (i.e., fingerprint). The general architecture of our access control system and the principle how the multi-factor authentication process works with smart phone is depicted in Figure 1.

Figure 1: Multi-factor authentication system using smart phone and embedded SAM module.

We refer to [2] for more details of AKA principles. We implemented applications for both Android phone and SAM module. The Algorithm 1 shows the cryptographic core of the access terminal (i.e., MultOS smart card) whereas Algorithm 2 shows the cryptographic core of the user device (i.e., Android phone).

Algorithm 1 AKA-SERVER-SIGNVERIFY(Y, sk_S, pk_C)

1: $r_S \in_R \mathbf{Z}_q^*$
2: $t_S \leftarrow g^{r_S}$
3: $e_S \leftarrow \mathcal{H}(Y, t_S)$
4: $s_S \leftarrow r_S - e_S \cdot sk_s \bmod q$
5: **Send:** $Y, \sigma = (e_S, s_S)$
$\triangleleft \triangleright$ AKA-CLIENT-PROOFVERIFY
6: **Receive:** $\pi = (e_C, s_C)$
7: $t' \leftarrow g^{s_C} \cdot pk_C^{e_C}$
8: $\kappa \leftarrow t'^{r_S}$
9: $\tau_S \leftarrow e_C \stackrel{?}{=} \mathcal{H}(Y, t', \kappa)$
10: **return** $\tau_S = 0/1, \kappa$

Algorithm 2 AKA-CLIENT-PROOFVERIFY(Y, σ, pk_S, sk_C)

1: $t'_S \leftarrow g^{s_S} \cdot pk_S^{e_S}$
2: $\tau_C \leftarrow e_S \stackrel{?}{=} \mathcal{H}(Y, t'_S)$
3: **if** $\tau_C = 0$ **then exit**
4: $r_C \in_R \mathbf{Z}_q^*$
5: $t \leftarrow g^{r_C}$
6: $\kappa \leftarrow t_S'^{r_C}$
7: $e_C \leftarrow \mathcal{H}(Y, t, \kappa)$
8: $s_{PIN} \leftarrow \mathcal{H}(PIN)$
9: $s_C \leftarrow r_C - e_C \cdot (sk_C + s_{PIN}) \bmod q$
10: **return** $\tau_C = 0/1, \pi = (e_C, s_C), \kappa$

Both algorithms need to perform modular and elliptic-curve operations, hash functions to authenticate communicating entities and establish common secret for future encryption of communication channel. In our implementation, we used secp256k1 elliptic curve, SHA-1 hash and 3DES-ECB cipher algorithms. In user application, one have to provide valid PIN code to be successfully authenticated. During authentication process a symmetric key is established. The key is used to encrypt user token sent from smart phone. The token contains time information, that can be used by terminal to clarify if the user has enough permissions and has been successfully authenticated by using another factor.

Testing terminal application for Raspberry Pi uses pcsc-lite external library, that makes it possible to create communication channel between any devices communicating via APDU messages. Since the Android Development Kit (ADK) tool does not provide any public elliptic-curve function calls, we used micro-ecc library to compute elliptic-curve operations. To do so, we implemented the core functions on smart phone in C++ using Android Native Development Kit (NDK). NDK allows us to archive much better performance results during cryptographic core execution comparing with Java libraries.

3. IMPLEMENTATIONS ASPECTS

Smart cards differ in processor performances, memory capacities and cryptographic support. The AKA protocol needs to use a secure hash function, a cipher algorithm, and perform operations over

elliptic-curve such as scalar point multiplication and point additions, as shown in Algorithm 1 and 2. However, different smart cards have different cryptographic support, and therefore, we have to select the most satisfying one, see [3] for more details. For example, JavaCard platform can offer smart card development environment based on Java programming language. These cards support wide range of secure cryptographic algorithms, however, they miss the support of modular arithmetic operations. BasicCard platform is based on Basic programming language. This platform offers a variety of algorithms, but is very limited in performance. MultOS platform can be programmed using C or Assembly language. All functions are provided by integrated framework. These cards show the best performance results and are considered very secure. In fact, they are used also in bank systems. Smart cards platform based on .NET framework called .NET Cards do not provide needed features, such as elliptic-curve operations. A detailed overview of cryptographic support across different smart card platforms is presented in Table I.

Table I: Smart card cryptographic support.

	JavaCard	**BasicCard**	**MultOS**	**.NET Card**
3DES	Yes	Yes	Yes	Yes
AES	Yes	Yes	Yes	Yes
DSA	Yes	No	No	No
SHA-1	Yes	Yes	Yes	Yes
SHA-2	Yes	Yes	Yes	Yes
SHA-3	Yes	No	No	No
MOD[1]	No	Yes	Yes	No
ECOP[2]	No	Yes	Yes	No

Note: [1]Modular arithmetics,[2]Elliptic-curve operations

In order to make our system more flexible, we optimized the APDU communication model. In particular, we use APDU encapsulation technique, which allows generating and inserting APDU command messages for the receiver to the APDU response message of the sender. Thanks to these techniques, the terminal does not need to know the communication protocol, it just simply forwards APDU messages between these two devices. The terminal starts with SELECT command message sent to the SAM module (i.e, smart card). This message is only for selecting the applet we want to use. Then the terminal waits for the smart phone to be attached. After that, SELECT command is transmitted to the smart phone, which responds with a data payload containing the APDU message and status code. The message in the data payload is then directly sent to the SAM by the terminal and the status code is removed. The SAM can also put a message to the data payload and send it back according to the status code. This communication method works indefinitely unless a specified status code is detected by the terminal. The status code represents the result of the authentication process.

4. EXPERIMENTAL RESULTS

The implemented experimental Android application contains of user interface, as one can see on the left side in Figure 2. A user can register to the application, reset any existing registration and turn on the authentication mode. The user registration requests the PIN code from the user. Then, the PIN code is required from the user to turn on the authentication mode. The mode is running for 30 s. If the user decides to clear all data, it is possible to trigger the reset process through the reset button. As an alternative to the PIN code, the user can use a fingerprint as a second authentication factor. This does not impact the cryptographic core of the authentication protocol. The difference is that the user is not required to insert the PIN code during the authentication phase. Note that the PIN code is stored in the application and is protected with the fingerprint of the user.

The development environment uses Raspberry Pi as common interface. The service for APDU transactions has its own output. If the application runs in debug mode, the log is printed to the terminal. The log of successful authentication of the user is depicted on the right side of Figure 2. The SAM module in the terminal determines the authentication result. If the authentication is successful, the SAM returns status code `0x9001`, otherwise it returns `0x9002`. The authentication result message carries

Figure 2: Smart phone application screen (left side) and log of successful authentication (right side).

an information with timestamp. The timestamp comes from smart phone and is encrypted using the symmetric key from AKA protocol. It serves to decide if the system can allow the user to enter a protected area in specific time period. Our benchmarks show how time consuming the implemented protocol is. The used smart card for verification part of the protocol is ML4-G17 model with MultOS version 4.2.1 and for user side we consider Huawei Nova 3 smart phone. The smart phone runs Kirin 970 processor clocked at 2.4 GHz with four ARM Cortex A73 cores and four ARM Cortex A53 cores. The protocol requires almost 2.6 s on SAM module and only 0.2 s on smart phone.

5. CONCLUSION

In this paper, we present our access control system using multi-factor authentication protocol, hardware terminal with embedded SAM module and clock, and Android smart phone application for users to prove their identities. The implementation is secure, not too computationally demanding and ready to run on IoT devices. In order to make the system more dynamic, flexible and extend usability of APDU communication model across various devices, we extended it with encapsulation method that makes possible to forward APDU messages between SAM and user's device without intervention from terminal. In the future, we are about to extend the system with addition peripheral devices, that user would be able to use as another authentication factor. Such device could be for example a smartwatch.

ACKNOWLEDGMENT

This research has been supported by the grant VJ01010084 of the Ministry of the Interior of the Czech Republic, "Digital evidence in criminal proceedings" in programme Impakt 1 (2022-2025).

REFERENCES

[1] European Parliament, Council of the European Union, "Directive (EU) 2015/2366 of the European Parliament and of the Council of 25 November 2015 on payment services in the internal market, amending Directives 2002/65/EC, 2009/110/EC and 2013/36/EU and Regulation (EU) No 1093/2010, and repealing Directive 2007/64/EC", 2015, https://eur-lex.europa.eu/legal-content/EN/ALL/?uri=celex:32015L2366.

[2] P. Dzrenda, S. Ricci, R. C. Marqués, J. Hajný, P. Číka, "Secret sharing-based authenticated key agreement protocol", The 16th International Conference on Availability, Reliability and Security, 2021, doi:10.1145/3465481.3470057.

[3] L. Malina, P. Dzurenda, J. Hajný, Z. Martinásek, "Assessment of Cryptography Support and Security on Programmable Smart Cards", The 41st International Conference on Telecommunications and Signal Processing (TSP), 2018, https://ieeexplore.ieee.org/document/8441334.