Generating Secure Session Keys from Shared Secret Information for Multi-Application IC-card Systems

Uddin, Mohammad Mesbah
Department of Computer Science and Communication Engineering Graduate School of Information Science and Electrical Engineering Kyushu University

Mori, Tatsuya
Department of Computer Science and Communication Engineering Graduate School of Information Science and Electrical Engineering Kyushu University

Yasuura, Hiroto
Department of Computer Science and Communication Engineering Graduate School of Information Science and Electrical Engineering Kyushu University

Inoue, Koji
PRESTO, Japan Science and Technology Agency | Department of Computer Science and Communication Engineering Graduate School of Information Science and Electrical Engineering Kyushu University

https://hdl.handle.net/2324/6235

Generating Secure Session Keys from Shared Secret Information for Multi- application IC-card Systems

Mohammad Mesbah UDDIN† Tatsuya MORI† Koji INOUE†‡ Hiroto YASUURA†

†Department of Computer Science and Communication Engineering
Graduate School of Information Science and Electrical Engineering
Kyushu University
6-1 Kasuga-Koen, Kasuga-City, Fukuoka 816-8580, Japan

‡ PRESTO, Japan Science and Technology Agency
4-1-8 Honcho Kawaguchi, Saitama 332-0012 Japan

E-mail: {mesbah, t-mori, inoue, yasuura}@c.csce.kyushu-u.ac.jp

Abstract

In multi-application IC card systems, when a user's card and an application provider's terminal interacts, it must be assured that only legitimate parties are allowed to initiate any corresponding application process, and that the initiators must not be replaced by others until the end of the application process. In this paper, we address the latter as a requirement for binding authentication with application execution, and discuss a protocol that provides the binding by using secure session keys generated from secret information shared by the corresponding parties.

Keywords: *Session key, IC-card systems, application security, mutual authentication.*

1 Introduction

With the rapid development of network and device technologies, computing environment is becoming pervasive. The advent of RF wireless technology, and other high performance SoCs (System on a Chip), introduces multi-application IC-card systems which enable us to use services from different vendors using contactless IC-cards as identifying devices. Although those systems make our daily life more convenient, the issue of security is of great concern. In a network society, where participants are always invisible, care must be taken to make sure that only legitimate parties can use any service.

Trusted third party (TTP) based scheme is being used by many systems to ensure that only eligible parties can use a service. In this scheme a user and/or a service provider asks the TTP through a secure channel to vouch for the counter party. Only after a successful authentication, a service can commence. However, in an IC-card system, due to the lack of secure channel and limited power-energy issues, it is less likely that every time to use a service, the participants will contact the third party. Rather, the participants would like to have a localized scheme of authentication [3]. In this scheme, the TTP provides some information to share between each pair of user and service provider, and they can verify their identity without contacting the TTP. Localized scheme has the advantage that it can be functioning even after earthquake, or similar catastrophes.

In a multi-application IC-card system, the usable power and enery is very limited. Therefore it is very important that authentication and authorization scheme must be "light". Service vendors (or, service providers) come up with different applications including database, e-commerce, authorization of utilities, and so on. While some of the applications may require highly secure authentication and authorization schemes, many others require simply "moderate" schemes. Although public key based schemes are highly secure, they are prone to computational cost. On the contrary, shared secret key based solutions can provide moderate security and use less computational power [1] [4].

However, if shared secret key based scheme is chosen, it becomes necessary to split authentication and application execution in two phases because of security reasons. It gives rise to the question *"how to propagate authentication to the exe-*

cution of application?.”. In other words, *”how do we ensure if the authenticated parties triggers the application?”*. This is a very important issue for a multi-application IC-card environment, because of the invisible nature of wireless communication — if the control of application execution of some legal user A is taken over by someone next to him, neither he can be seen, nor he can be recognized. In this paper, we proposed a protocol to generate session keys as a solution to the problem. To generate the session keys, the protocol uses secret information that is provided by the TTP.

The rest of the paper is organized as follows. Section 2 outlines the multi-application IC-card system and communication model. Section 3 describes security requirements for the system. Section 4 shows our porposed scheme and section 5 concludes.

2 Multi-application IC-card System

2.1 The System

In this section, we describe the model of a multi-application IC-card system (Figure 1). The system consists of three potential participants – Users, Service Providers, and the Issuer.

A user is a human being who uses some of the services. A user accesses a service application by using an IC-card. Therefore, each user is associated with an IC-card. An IC-card typically contain a full microprocessor, memories and optionally some cryptographic modules. Most IC-cards are equipped with a serial interface (the metallic contacts) but increasingly they are equipped with a RF-interface (contactless IC-cards). In this paper, we focus on contactless IC-cards.

Service providers deal with services to users; they provide applications to be used by the users. A service provider uses IC-card readers and servers to deal with services to the users. A service provider may use multiple IC-card readers to serve the users at different sites. Each service provider is associated with a set of IC-card readers and a server system. Unlike the systems where the card reader is used to verify the card’s identity, in our system the card reader simply passes data between the IC-card and the SP’s server in a recognizable format.

The issuer is a a trusted organization (community, company, school, credit card service companie, etc) that has the duty to protect the users and service providers. The issuer is responsible for vetting the users and service providers and for arranging a mechanism for vouching their eligibility to the other participants where necessary. Therefore, the issuer needs to maintain vetting systems, databases, vouching systems, etc – together they are referred to as the issuer’s systems.

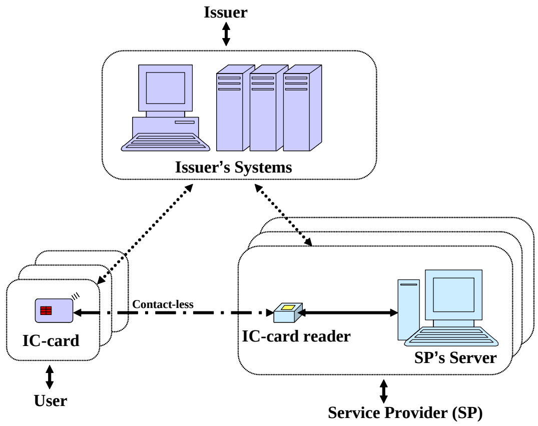

Figure 1. A multi-application IC-card system.

2.2 Issuing of Secret Information

In a multi-application IC-card system, it must be assured that a service can be carried out only by and only between legitimate parties. The issuer ensures the ”legitimacy” of a user and a corresponding service provider by issuing some secret information (secret key) to them. In this paper, we assume that the secret information is shared between the user and the corresponding service provider. Operations necessary for issuing secret informations are carried out by secure methods and through secure channels. Services can commence only after issuing of the secret information being complete.

In order to obtain the secret information, users and service providers must go through the following registration procedure:

Step1: A user requests issuing of an IC-card to the issuer.

Step2: The issuer examines a user’s personal identity. If the user verification is okay, the issuer personalizes an IC-card (i.e., writes secret informations, data, program, etc. in the IC-card). The IC-card is then given to the user by a secure method. User’s data, secret information is stored in a secure database.

Step3: A service provider that wants to offer service(s) to users, asks to the issuer for permission.

Step4: The permission is ensured by the issuer upon vetting the service provider. Typically, the issuer gives the service provider a list of secret information, data, and program, etc. for each corresponding user.

Step5: The user is notified about the correspondence between the secret informations and the service providers. At this stage, a user and a corresponding service provider shares some secret information.

Step6: Using the shared secret information as a method of identification, a user and a corresponding service provider can establish a service. Note that a user usually operates offline with the issuer, whereas the service provider may be online or offline with the issuer.

2.3 Communication Model

In an IC-card system the communication is direct (peer-to-peer, or P2P). For sending and receiving messages, two primitives, respectively *send(message)* and *receive(message)* are provided.

If a user and a service provider wants to carry out a service, they must do it by sending messages to and receiving messages from each other; that is, a communication link must exist between them. Our focus here is not with the link's physical implementation, but rather with the issues of it's logical implementation, such as it's logical properties:

- A link is automatically established between every pair of entities that want to communicate. The entities need to know only how to refer the other during each communication.
- Between each pair of entities, there exist exactly one link.
- The link is bidirectional.
- A link is assosiated with exactly two entities at the same time.
- The link is of Zero Capacity, i.e., the link cannot have any messages waiting in it.

3 Security of Services and Propagation of Authentication

3.1 Security of Services

In an IC-card system, where the communication channel is insecure, hostile parties are present, it must be assured that no illegal party can use a service (i.e., execute an application). That is,

- A user must be able to identify the legitimacy of a service provider and vice-versa (mutual authentication).
- Only the verified parties are allowed to execute application (propagation of authentication).

In order to be able to use any application, there must exist a communication link between a user and the corresponding service provider. When the communication is established, we say that they are in connection, or in a session. Within a session, a user and a service provider goes through a mutual authentication procedure followed by exucution some of application process. Protocols for mutual authentication is proposed in [4]. After mutual authentication succeeds, the participants switch to executing the application process. Since the communication channel is insecure,

- sensitive data must encrypted over the channel (encryption), and
- same encrypted data is not used multiple times (unlinkability)

within a session.

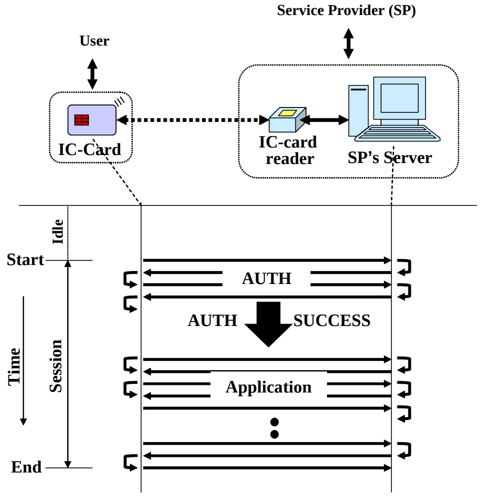

Figure 2. Interaction between a user and a service provider during a session.

In a multi-application IC-card system, thousands of users connect to service providers using invisible, wireless connection. Eavesdropping, recording, or modifying of data, impersonating, or accessing applications becomes very easy and sometimes untrackable. Therefore it is very important to ensure that a service can take place only between authenticated entities.

3.2 Public Key And Shared Key Based Approaches

In order to ensure that application is executed by the authenticated entities, public key based schemes may be adopted. In this scheme, authentication is distributed throughout a session. Let us consider an example where a user Alice and a service provider Bob wants to do business. Alice has private (or, secret) key SK_A and public key PK_A, whereas Bob has private key SK_B and public key PK_B. After mutual authentification is successful, Alice can use Bob's public key PK_B to encrypt M and then sends the encrypted message to Bob. Bob decrypts the message using his private key SK_B to obtain M. Bob can use a similar method to send a message to Alice.

Although public key based scheme is cryptgraphically strong, in a multi-application IC-card environment, where computational resources are limited and computational cost is a key factor, using public key may not be cost and resource effective.

Another possible approach is to use shared secret key based method. In such a method, the same secret key maynot be used to encrypt every message in a session since it violates unlinkability. Moreover, if shared secret key based scheme is chosen, it becomes necessary to split authentication and application execution in two phases because of security reasons. One main reason is that the secret key of some participant may not be allowed to be accessed by a corresponding party, because some unwanted program viruses may leak the secret. When authentication and application execution are split, it is important to propagate authentication to the application process, that is, to ensure that the authenticated parties can trigger any application and continue execution.

3.3 Propagation of Auhentication

If mutual authentication is used for verifying user and service providers, *"how do we ensure that any of the initiator parties are not replaced during switching from authentication to execution of application?"*. We refer to this problem/attack as application take over (Figure 3).

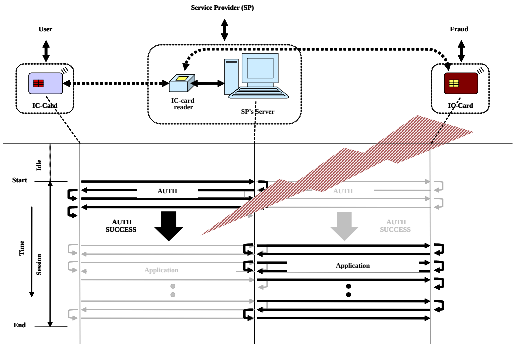

Figure 3. Taking over of a user application by a hostile party.

After authentication, the participants can chose not to use any encryption. If they donot use any encryption, then it is easy to take over the application. Alternatively, they can encrypt every message during the session using the secret key. If secret key is used many times for encryption, the secret key may become breakable, resulting in an ultimate damage of the system. Once, the secret key is broken, taking over an application becomes easy.

Therefore, we need a way to propagate the authentication. In section 4 we propose a scheme to generate session keys using shared secret keys as a method of prpagating authentication to application execution.

4 Generation of Secure Session Keys Using Shared Secret Information

4.1 Basic Concept

In order to propagate authentication to application execution, we need to bind authentication and application execution. Binding is done using generating session keys and then propagating those keys securely to the execution unit of the participatants. Generation of keys is done by using shared information such as shared random number genarator function. Nevertheless, within a session, it is necessary to send the generated keys over the insecure channel. The shared secret key is used to encrypt the session key. Upon receiving an encrypted session key, the recipient can decrypt it using it's own shared secret key. Although the same shared key is used in different sessions for encrypting session keys, encryption of random values makes it safe in terms of unlinkability. Generated session key is passed to the application execution securely by some shared memory or message passing protocols inside each of the parties.

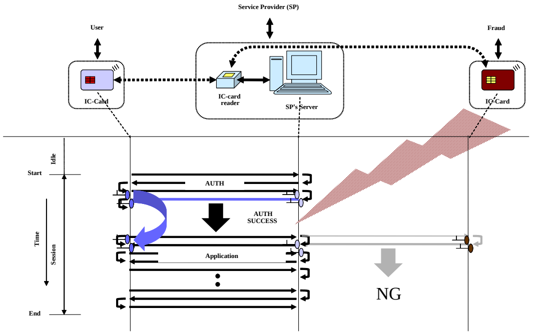

Figure 4. Binding authentication with application execution.

4.2 Generation of Secure Session Keys

Below we state a protocol for generating and exchanging session keys between two entities, say Alice and Bob, using shared secret information. Initially, Alice and bob shares a secret key SK, hash function H, random number generating function F_{rand}, and a function F_{inc} to update/increment session keys. They also know the mechanism of addressing the corresponding parties using a mapping function f. Choice of the seed $S_{F_{rand}}$ to F_{rand} is arbitrary. The authentication protocol uses on one-way hash function.

Step 1: Alice and Bob exchanges information for authentication.

Step 2: Bob confirms Alice's identity, and sends some more information to reveal Bob's identity to Alice, along with $E_{SK}(r_0)$, where $r_0 = F_{rand}(S_F)$, a random number that is not sent through the channel raw.

Step 3: Upon receiving the message, Alice confirms Bob's identity, and decyphers the aditional message, $D_{SK}(E_{SK}(r_0))$ to obtain r_0. Alice sends acknowledgment to Bob and passes r_0 to the application process to be used as a session key.

Step 4: Upon receiving the acknowledgement, Bob passes r_0 to the application process to be used as a session key.

Step 5: Using r_0 as a session key, Alice and Bob can now start executing an application. Within the application, r_0 can be incremented for each new message to be sent over the channel – for each new message M_{new}, the key for encrypting the message is $r_{new} = F_{inc}(r_{prev})$. That is for a sequence of messages $M_1, M_2, \ldots, M_n$, the keys for that session are $r_0, F_{inc}(r_0), \ldots, F_{inc}^{n-1}(r_0)$. Since the keys are generated each time it is necessary, it gives security like a one-time passord system.

4.3 Discussion

In this section, we discuss the correctness of our protocol and possible attacks. Letting Alice initiate the negotiation process, actions in the protocol is illustrated in Figure 5.

In our protocol, mutual authentication is achieved by using a one-way hash function based scheme proposed in [4]. According to the scheme, authentication is known after the comparison of action 8 (Figure 5). At this stage, after Alice confirms it is a message from Bob, Alice retrieves r_0 using the secret key SK. Then, Alice sends a confirmation message (action 9), and pass r_0 to the application execution unit (action 10). After Bob receives confirmation from Alice, Bob sends r_0 to the application execution unit. At this stage, only Alice and Bob has the shared information r_0. Using this as a session key they can start sending messages and execute the application. For each new message to be sent, or received, the secret key is updated using an updating function F_{inc}. Since, updating is done at both the ends, Alice and Bob always shares the same key. Because the keys are changed for every message in a session, it provides strong cryptographic scheme similar to one-time passwords.

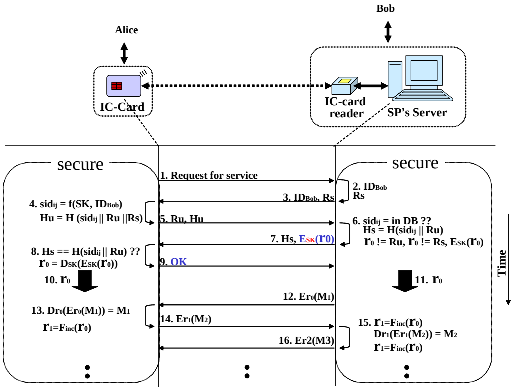
Alice
Bob
IC-Card
IC-card reader
SP's Server
secure
secure
1. Request for service
2. ID_Bob Rs
3. ID_Bob, Rs
4. sid_ij = f(SK, ID_Bob) Hu = H (sid_ij || Ru ||Rs)
5. Ru, Hu
6. sid_ij = in DB ?? Hs = H(sid_ij || Ru) r_0 != Ru, r_0 != Rs, E_SK(r_0)
7. Hs, E_SK(r0)
8. Hs == H(sid_ij || Ru) ?? r_0 = D_SK(E_SK(r_0))
9. OK
10. r_0
11. r_0
12. Er_0(M_1)
13. Dr_0(Er_0(M_1)) = M_1 r_1=F_inc(r_0)
14. Er_1(M_2)
15. r_1=F_inc(r_0) Dr_1(Er_1(M_2)) = M_2 r_1=F_inc(r_0)
16. Er2(M3)
Time

Figure 5. Actions in the secure session key protocol (X||Y: combined value of X and Y, H(X): hashed value of X).

In order to investigate possible attacks, let Eve was succesful recording the conversation between Alice and Bob. She may modify the message, or generate her own message and send it to Alice and/or Bob. Eve can have the following information: ID_{Bob}, R_u, R_s, H_u, H_s, $E_{SK}(r_0)$, OK, $E_{r_i}(M_{i+1})$ (where, i = 0, ... Given the above information, is it possible for her to determine r_0 or other r_i's, SK, and F_{inc}?

In order to determine r_0 Eve must know either (1) about the random number generator F_{rand} and it's seed $S_{F_{rand}}$ used for a session, or (2) SK and decrypting function D. Since F_{rand}, rules for $S_{F_{rand}}$ is passed to Alice and Bob using a secure channel (by the issuer), it is unlikely that Eve knows about them; knowing SK and D would be similarly impossible. Therefore, determining r_0 would require brute-force method that is considered computationally infeasible. However, Eve may attempt some adversary to guess about r_0. Even if eve succeeds determining r_0, it would do little for her as the session keys are updated for each message. In order to use r_0 effectively, she must also know the updating function F_{inc}.

For determining the updating function F_{inc}, Eve may record all the conversations between Bob and Alice, and may succeed in analyzing the function. For applications that need large number of message exchanges, F_{inc} must be chosen carefully.

Yet, another possibility for Eve is to attempt to determine SK using information of several sessions between Alice and Bob. Although it depends much on the type of random numbers and their encrypted values, in general, with all r_0's being random, the outputs would generate random values. Thus predicting about SK would be computationally infeasible.

Alternatively, Eve may also try to modify messages. If the message in action 7 is modified, it is highly likely that authentication will fail and the session will stop. However, if only the $E_{SK}(r_0)$ part is modified, then, Alice would produce a session key that is different from Bob's, and won't be able to use the application. Eve won't be able to use the application, either. However, Bob would notice that something has gone wrong.

Eve may also try to attack the acknowledgement "OK" (action 9). If she intercepts "OK" and then sends it to Bob without modifying, then Bob executes step 11. In this case, Eve cannot start executing any application without knowing r_0. However, if Eve modifies "OK" and send it to Bob, then Bob notice the modification and stops proceeding any further.

Eve may want to generate acknowledgement by herself and send it to Bob. If Eve's message "OK" reaches Bob prior to Alice's "OK", and Bob don't notice it, Bob will execute step 11. Even in this case, Eve won't be able to start executing the application without knowing r_0. In case that alice rejects Bob's authorization, the session terminates. Eve cannot, as in other cases, start any application process withot knowing r_0.

5 Conclusion

In this paper, we have discussed about securing services for multi-application IC-card systems. We have considered localized mutual authentication between two parties to be very prospective for those systes. Therefore, we have considered secret sharing methods for enabling localization of authentication process. Due to the limitation of battery capacity (energy) and other computational resources in IC-card, lightweight protocols are indeed essential. Moreover, many applications may need security only up to a standard level. From these two points, we have considered about separating authentication and application execution phases within a session. However, in doing so, a protocol for propagating authentication effect to the application execution phase is proposed.

In this paper, we have outlined the basic concepts and usability of our scheme. More detailed analysis and comparisons will follow in a later work. Future work also include considering secure incremental functions and secure signatures over a session.

Acknowledgement

We are thankful to the members of System LSI Research Center of Kyushu University for their valuable suggestions and co-operation while preparing this article. This work has been supported by the Grant-in-Aid for Creative Scientific Research No.14GS0218 of the Ministry of Education, Science, Sports and Culture(MEXT) from 2002 to 2006. We are grateful for their support.

References

[1] Hiroto Yasuura, "Towards the Digitally Named World -Challenges for New Social Infrastructures based on Information Technologies-", Proceedings of Euromicro Symposium on Digital System Design - Architectures, Methods and Tools-(DSD2003), pp.17-22, Sep. 2003.

[2] Hiroto Yasuura, "[Plenary Speech 2p.1]Digitally Named World: Challenges for New Social Infrastructures", 5th International Symposium on Quality Electronic Design(ISQED 2004), pp.323, Mar. 2004.

[3] Yoichiro Hamasaki and Hiroto Yasuura, "A Proposal of Secure Infomation Infrastructure based on PID (in japanese)", Research Reports on Information Science and Electrical Engineering of Kyushu University, Vol.7, No.2, pp.139-148, Sep. 2002.

[4] Yasunobu Nohara, Sozo Inoue, and Hiroto Yasuura, "Toward unlinkable ID Management for Multi-service Environments",Proc. of PerCom 2005 Workshops, pp.115-119, Mar. 2005.

[5] Bruce Schneier, "Applied Cryptography", John Wiley and Sons, Inc.

[6] B. Schneier, A. Shostack" Breaking Up Is Hard To Do: Modeling Security Threats for Smart Cards ", USENIX Workshop on Smart Card Technology, USENIX Press, 1999, pp. 175-185.

[7] NIST, "Federal Information Processing Standard Publication 180-2", Aug. 1, 2002.

[8] Jennifer G. Steiner, Clifford Neuman, and Jeffrey I. Schiller. Kerberos: An authentication service for open network systems. In Proceedings of the Winter 1988 USENIX Conference. USENIX, February 1988.