SRS-Based Automatic Secure Device Pairing on Audio Channels

Young Sam Kim[1,2], Seung Hyun Kim[2], Seung Hun Jin[1,2]
University of Science and Technology[1], Daejeon, Korea
ETRI[2], Daejeon, Korea
kim03, ayo, jinsh{@etri.re.kr}

Abstract

Secure device pairing is necessary to communicate securely between mobile devices. Although several researches have been conducting, it was hard to prevent MITM attack on the wireless communication channel. The reason is that because secure device pairing assumes that participants in communication have never met before, they cannot share some information for detecting MITM attack before starting communication. SAS-based AKA(Authenticated Key Agreement) protocol could prevent MITM attack efficiently using an authenticated channel(e.g. OOB channel). The later researches are focused on improving usability than security or efficiency of a protocol. The reason is that authenticated channel in SAS-based AKA protocol needs user's help, so unintended failures of key agreement can be arisen. Automatic pairing can solve the problem but related research is not abundant. In this paper, we propose a key agreement protocol using SRS(short random string). We also implement the automatic secure device pairing protocol based on audio channel and analyze usability and security.

1. Introduction

Mobile security is becoming more important as mobile devices like PDAs, smart phones, laptops are generally used. Secure device pairing is one of the important technologies to assure the confidentiality on the wireless network. Public key based D-H KAS(Key Agreement Scheme) is the most widely used method for secure device pairing. D-H KAS makes to agree a key possible between strangers based on DLP(Discrete Logarithm Problem).

However, MITM attack on D-H KAS is possible not because DLP but vulnerability of the protocol. Several researches have been conducted to prevent the MITM attack. The results consist of two ways. One is that both communicators share a short secret through an authenticated channel and proceed D-H KAS. The other is that both communicators proceed KAS and then they generate SAS(Short Authenticated Strings) message respectively using parameters in KAS and compare whether two SAS messages are same or not. These two methods can prevent MITM attack but it needs an authenticated channel which is impossible MITM attack. Most researches in the past require user's help to implement the authenticated channel. They assume physical channels related to users(e.g. sight, hearing, action etc.) as an authenticated channel. As a result, key agreement becomes secure but we have to consider usability. The recent researches focus on the usable security which improve the usability. Most of them require user's help like comparison numbers[8], melodies, sentences[12], or shaking devices[1,14] etc.

According to the [9], most users prefer automated comparison to manual comparison when they pair the devices. The [6] and [9] deal with automatic pairing. However, they have some limitations. The [6] has long period for pairing and it's hard to adopt for constrained devices(e.g. devices with no display). The [9] require the ATD(Auxiliary Third Device) that is not practical.

In this paper, we propose automatic secure device pairing on audio channels. Our contributions are as follow:

1. We propose SRS-based key agreement protocol. This protocol doesn't require comparison by users while SAS-based AKA require it. In addition, SRS-based key agreement protocol doesn't require typing while MANA(Manual Authentication)[2] protocol does. Therefore, our proposed protocol can pair two devices automatically.
2. To implement our protocol, we don't need auxiliary device like ATD.
3. We use audio channels as an authenticated channel, so it can be used on the constrained devices.
4. Our implementation is quite fast compared with a result of other researches.

This paper consists of 7 chapters. In chapter 2, we overview the related researches. In chapter 3, we describe proposed protocol. In chapter 4, we describe how to implement the protocol using audio channels and analyze the usability. Next, we discuss about our result of implementation in chapter 5. In chapter 6, we analyze the security of our protocol and finally conclude in chapter 7.

2. Related Works

In the recent researches, they assume device's I/O capabilities: audio, video camera, camera, LED, button, accelerator and so on as authenticated channels. At first, we look into SAS-based AKA [4,5] protocol that is used in many researches. Next, we examine Saxena et. al.'s researches[6,9] dealing with automatic secure device pairing.

2.1. SAS(Short Authenticated Strings)-based AKA protocol[4,5]

Vaudenay proposed the protocol(Figure 1). When two devices *A* and *B* want to pair each other, *A* and *B* exchange their public keys and random strings. After that, they generate the SAS and SAS' messages respectively and exchange them on an authenticated channel. *B* generates SAS' using his/her public key, random string, and *A*'s random string. *A* generates SAS using *B*'s public key and random string, and his/her random string. Now each device can compare SAS and SAS' messages. If they are the same then device A and B agree a key. Otherwise, protocol fails.

To MITM attack, an attacker must send his public key to *A* and *B*. In this case, SAS and SAS' messages will be different, so we can detect the MITM attack.

In SAS-based AKA protocol, there are three messages exchanged on the wireless channel. It is called commitment protocol or 3-move AKA protocol. If participants just exchange their public keys and random strings, an attacker can manipulate the values. Thus, the MITM attack becomes possible. The MITM attack also becomes possible though participants use the commitment protocol. Because if k value in the Figure 1 is not sufficiently large, an attacker can know messages of commitment. Therefore, we should consider above possibility of attacks when implementing SAS-based AKA protocol.

SAS-based AKA protocol is very simple and secure. After presenting the protocol, many researches have conducted related usability and we will examine two of them by Saxena et. al. The reason is that those researches implement automatic secure device pairing like us.

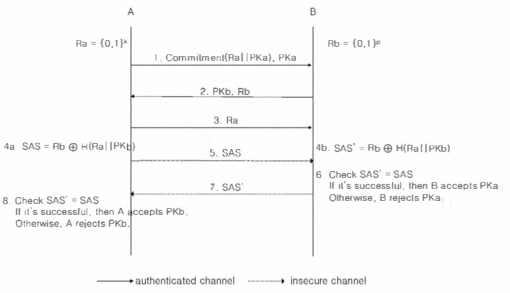

Figure 1. SAS-based AKA protocol

2.2. Automatic Secure Device Pairing on Visual Channel[6]

Saxena et. al. proposed automatic secure device pairing using a video camera and a LED as an authenticated channel. One device represents SAS messages using a LED's on/off, then other device records and analyzes them using a video camera. It implements automatic secure device pairing but there are some limitations. First, for pairing a user should shoot a LED for quite a long time. Second, it's not usable for constrained devices because it requires devices with display capability.

2.3. Automatic Secure Device Pairing using beep or blinking[9]

Prasad and Saxena implemented authenticated channel using beep and blinking[10]. These researches are significance because they presented new ways to implement an authenticated channel. However, the usability is not good. The completion time of pairing is about 20 seconds and it also needs manual comparison, i.e. user's help. Users have to check that beep or blinking from both devices is identical or not. This process is heavy burden to users as a result fatal error[1] rate becomes higher. To solve this problem, Sexena et. al.[9] proposed automatic secure device pairing. They use an auxiliary device named ATD that substitutes a user's role and makes fatal error rate zero. But this research has a limitation. The thing is that ATD is not applicable in practice.

In this research, one of the meaningful results about usability is that 80% users prefer automatic pairing to manual pairing. This makes our goal more clearly.

3. SRS-based Key Agreement Protocol

Our proposed protocol uses the commitment protocol. SAS-based AKA protocol is very simple and also secure protocol, but there are some possibilities to improve. First, SAS-based AKA protocol is secure enough though we don't use *B*'s random string. The values that an attacker can manipulate against SAS messages are *B*'s random string and public key. And the value that an attacker can manipulate against SAS' messages is only *A*'s random string. To make SAS and SAS' messages identical, an attacker must discover *A*'s random string in step 1. Then an attacker can generate SAS message and calculate another random string which makes SAS and SAS' messages identical. But it's so hard because of commitment protocol. On the

[1] The interaction with a user is necessary for secure device pairing. However, a user cannot act according to the intended protocol. Failures of pairing from that reason are called "fatal error".

commitment protocol, nobody can discover committed value before knowing open value. Second, exchanging SAS and SAS' messages is an overhead for secure device pairing because the authenticated channel is face-to-face.

According to our analysis above, we propose SRS-based protocol(Figure 2) simpler than SAS-based AKA protocol. We use commitment protocol. When generating commitment we use short random string called SRS. Then, we transfer the SRS on the authenticated channel. Now a device *A* can verify commitment's integrity and finally authenticates *B*'s public key. In addition, a device *A* can know that a device *B* has received his/her public key correctly.

The protocol between *A* and *B* is as follow.

1. A device *A* sends his/her public key PKa to a device *B*.

2. A device *B* generates two random strings, R, and SRS where $SRS \xleftarrow{R} \{0,1\}^p$, $R \xleftarrow{R} \{0,1\}^k$. Then a device *B* calculates a commitment using PKa, PKb, SRS and R and sends the commitment to a device *A*.

3. A device *B* sends SRS to a device *A* through an authenticated channel.

4. A device *B* sends R to a device *A*.

5. A device *A* verifies the commitment received in step 2 using PKa, PKb, SRS, and R. If it is successful, a device *A* generates agreed key using authenticated *B*'s public key.

In our protocol, the MITM attack will be prevented because an attacker cannot impersonate *B*'s public key.

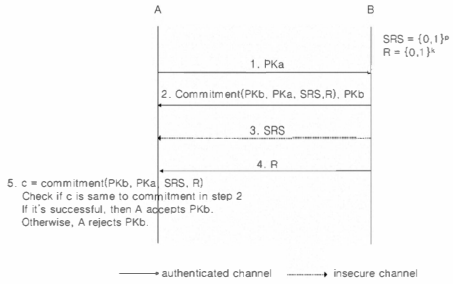

Figure 2. SRS-based KA protocol

But an attacker can impersonate *A*'s public key because the verification process will only be done in the device *A*. This attack is prevented when the pairing protocol proceeds between humans who own the device. When it's not under the human-to-human circumstance, the attack is also prevented by sending the result of verification from the device *A* to *B*.

Our SRS-based KA(Key Agreement) protocol is 3-move protocol like SAS-based AKA protocol. In proposed protocol, the number of messages is smaller than SAS-based AKA protocol as omitting A's random string. In addition, the number of operations is also smaller.

4. Implementation and Usability

4.1. Implementation

We implement SRS-based KA protocol on audio channels. We use a PC and a laptop as a device A and a device B respectively. That is, a microphone in a PC and a speaker in a laptop are used for establishment of authenticated channel. We focus on implementation about authenticated channel that SRS message is transferred.

For implementation, we use MFC as a programming language, Visual Studio 6.0 as an IDE, SHS-100V headset as a microphone, and a laptop's built-in speaker as a speaker.

The core of implementation is a way that sends SRS message through an authenticated channel. We sends SRS message using beeps with various frequencies and durations.

4.1.1. Beeps with various frequencies and durations. Data transmission using sound is easily affected by noise. It means that it's hard to transmit sophisticated data using sound.

We found out the way to make it possible. The idea is to use relative values. First, we choose beeps having frequency and duration like table 1. Then we select one as a standard beep(Medium) and select five beeps as SRS message. These five beeps are divided one of the Low, Medium, and High values by standard beep. Therefore we can transfer exact SRS message even if we use different audio devices corresponding different mobile devices.

	Low	Medium	High
Frequency(Hz)	500	1000	2000
Duration(ms)	200	400	600

Table 1. Beeps with three different frequencies and three durations for our implementation

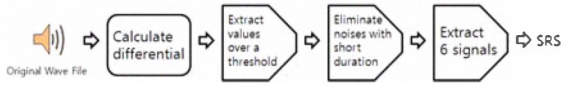

Figure 3. The process of extracting SRS from WAVE file

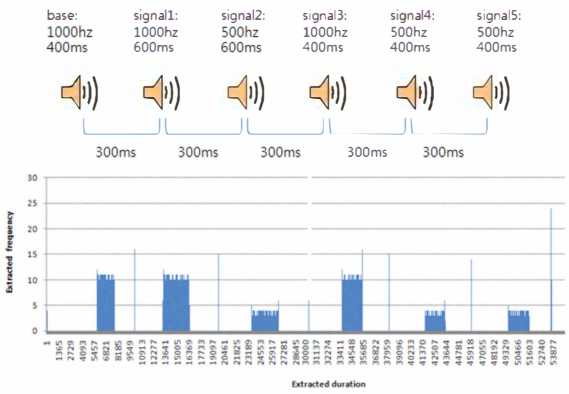

Figure 4. An example of beep and corresponding extracted 6 signals

We implement message transfer on audio channels based on the procedure in the Figure 3. To prevent interference between beeps we insert an interval, 300ms. Thus, transfer time for six beeps is at most 300ms *6 + 400ms + 600ms*5 = 5.2 seconds. We add some interval at first and end because of robustness. Consequently, our total completion-time for pairing is 6+α seconds(α is a sum of transmission time for messages on the wireless channel). According to the investigation by Kumar et. al.[15], the way that has the fastest completion time for pairing is numeric comparison and its time is 5~6 seconds. So we can say that our implementation is quite fast.

4.2. Usability

There is no user's help in our automatic secure device pairing, so we won't consider the fatal error. We only focus on the safe error.

We consider following two parts.

1. safe error rate
2. variation of safe error rate depending on the distance

Our experiment conducted under the silence environment(The problem of noise filtering is out of scope).

For the number 1, we executed our program 50 times and checked whether SRS message extracted from a WAVE file is correct or not(distance = 10cm). Corresponding result is on the table 2.

For the number 2, we recorded beep sound varying distance(5cm, 10cm, 20cm, 30cm, and 40cm) between a microphone and a speaker. Our recording was iterated ten times in each distance and the result is like Figure 5.

Table 2. Experimental results on safe errors

#success	#failure	Safe error rate(%)
50	0	0

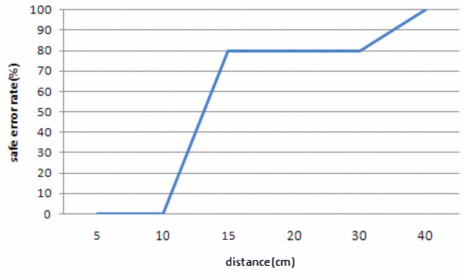

Figure 5. Graph for variation of safe error rate depending on the distance

5. Discussion

There are some limitations on the transmission of SRS message using audio channel. First, it's easily affected from noise. If we implement the protocol under the noisy environment safe error rate will be highly increased because of superposition and interference of sound. Second, the distance of data transmission depends on the device, especially a microphone. In our results of experiment, safe error rate starts to rapidly increase over the 10cm. We thought of our headset that we used in the experiment as the main reason. The distance of delivery of sound depends on a medium. In the experiment, we heard beeps over 1 meter. It means that the results about safe error rate depending on the distance are from hearing ability of headset.

According to the limitations above, when our implementation using audio channel is used, we have to keep short distance. If we use small mobile devices like mobile phones, it can be possible scenario.

By the way, the pairing limited short distance has some merits. First, it's insensitive to noise. One of the weak points about transmission by sound is that it's hard to use in the noisy environment. However, if the distance of delivery of sound is short, the affection from noise is also becoming short. Consequently, it can be insensitive to the noise from a little longer distance. Second, DoS attack is becoming harder. DoS(Denial of Service) attack is possible while MITM attack is impossible on an authenticated channel. But in our implementation, DoS attack is limited in the very short range. Thus, our implementation is reasonably resistant to DoS attack.

6. Security

We analyze the security of our protocol using the adversarial model in [5]. According to this adversarial model, an attacker has full controls on the wireless channel. An attacker can drop, delay, replay, and modify the whole messages on the wireless channel. However, an attacker cannot modify the

messages on the authenticated channel while he/she can have any other controls. An attacker can also simulate commitment value using the message he/she choose.

Theorem. Let *M* be an attacker followed by adversarial model. If the commitment value is generated using ideal hash function h : $\{0,1\}^m \rightarrow \{0,1\}^n$ and the length of SRS message is $p(\leq n/2)$, then the success probability of attack is less than or equal to 2^{-p}.

Proof sketch. By adversarial model an attacker can manipulate the messages in step 2 and 4 in our protocol. Therefore, we can consider the three possible attacks like below.

Case 1. Simply Guessing SRS message : Suppose that an attacker *M* drops the message in step 2, and sends his/her commitment *c'* generated by his/her public key PKm, arbitrary SRS', R', and PKa received from a device *A*. The attack is successful if and only if SRS and SRS' messages are exactly the same. This probability depends on the length of SRS message, so the success probability of attack is 2^{-p}.

Case 2. Discovery of correct SRS message from commitment : If an attacker *M* discovers the correct SRS message from correct commitment *c*, then he/she can derive correct verification of *c'* made by his/her public key PKm, random value R' and PKa received from a device *A*. By the assumption, our commitment scheme is ideal hash function. Thus, the security of commitment is $2^{-n/2}$ by the pre-image resistance.

Case 3. Leading to correct verification using incorrect SRS' message : Suppose that an attacker *M* drops the message in step 2, and sends his/her commitment *c'* generated by his/her public key PKm, arbitrary SRS', and R'. In step 3, an attacker can eavesdrop the correct SRS message. As a result, an attacker M can find R'' satisfied with the equation below.

h(PKm, PKa, SRS', R') = h(PKm, PKa, SRS, R'')

The probability of finding R'' is $2^{-n/2}$ by the 2^{nd} pre-image resistance.

By the assumption, the case 1's success probability is the largest.

In the [4,5], when the best way to attack on the SAS-based AKA protocol is just guessing, it requires 15-bit SAS message length for the security $3*10^{-4}$. According to our theorem, guessing attack on the SRS-based KA protocol is the best way. Thus, the SRS-based KA protocol is secure if the length of SRS message is 15 bits. We used 9^5-bit SRS message in the implementation, so we can say that our implementation is secure.

7. Conclusion

In this paper, we proposed SRS-based KA protocol and implement automatic pairing using a microphone and a speaker. SRS-based KA protocol is simpler than another KA protocols(e.g. SAS-based AKA, MANA, etc) and makes automatic device pairing possible. Automatic device pairing doesn't require user's help so we can eliminate the fatal error. And it also shortens the completion time of pairing. Therefore, we can improve the usability of device pairing.

We also tested safe error rate about our implementation using audio channel. As a result, we found out that safe error rate depends on the distance between a microphone and a speaker and especially safe error rate is 0% within particular distance. That is, both safe error rate and fatal error rate are 0% within particular distance.

Plus, the implementation using audio channels can be applied to constrained devices

8. Acknowledgement

This work was supported by the IT R&D program of MKE/KEIT [10035219, Smart Wallet for Mobile Identity Protection and Privacy].

9. References

[1] Holmquist, L.E., Mattern, F., Schiele, B., Alahuhta, P., Beigl, M., Gellersen, and H.-W., "Smart-its friends: A technique for users to easily establish connections between smart artefacts", UbiComp2001on Ubiquitous Computing, Atlanta, Georgia, USA, pp. 116–122.

[2] C. Gehrmann, C. J. Mitchell, and K. Nyberg, "Manual authentication for wireless devices", RSA CryptoBytes, 2004. vol. 7, no. 1,

[3] Jonathan M. McCune, Adrian Perrig, and Michael K. Reiter, "Seeing-is-Believing Using Camera Phones for Human-Verifiable Authentication", Proceedings- IEEE Symposium on Security and Privacy, 2005, pp. 110-124.

[4] Serge Vaudenay, "Secure communications over insecure channels based on short authenticated strings", Conference- CRYPTO, 2005, pp. 309-326.

[5] Sylvain Pasini and Serge Vaudenay, "SAS-based Authenticated Key Agreement", Conference- PKC, 2006, pp. 395-409.

[6] Nitesh Saxena, Jan-Erik Ekberg, Kari Kostiainen, and N. Asokan, "Secure Device Pairing based on a Visual Channel", IEEE Symposium on Security and Privacy, pp. 306-313, 2006

[7] C. Soriente, G. Tsudik, and E. Uzun, "BEDA: Button-Enabled Device Association," International Workshop on Security for Spontaneous Interaction (IWSSI), 2007.

[8] Ersin Uzun, Kristiina Karvonen, and N. Asokan, "Usability Analysis of Secure Pairing Methods", USEC, 2007, pp. 307-324.

[9] Nitesh Saxena, Md. Borhan Uddin, and Jonathan Voris, "Universal Device Pairing using an Auxiliary Device", Symposium on Usable Privacy and Security, USA, 2008.

[10] Ramnath Prasad and Nitesh Saxena, "Efficient Device Pairing using "Human-Comparable " Synchronized Audiovisual Patterns", Proceedings of the 6th international conference on Applied cryptography and network security, 2008, pp. 328-345.

[11] Volker Roth, Wolfgang Polak, Eleanor Rieffel, and Thea Turner, "Simple and effective defense against evil twin access points", Proceedings of the first ACM conference on Wireless network security, 2008, pp. 220-235.

[12] C. Soriente, G. Tsudik, and E. Uzun, "HAPADEP: human-assisted pure audio device pairing," in Information Security, 2008, pp. 385–400.

[13] Tim Kindberg, James Mitchell, Jim Grimmett, Chris Bevan, and Eamonn O'Neil, "Authenticating Public Wireless Networks with Physical Evidence", IEEE International Conference on Wireless and Mobile Computing, Networking and Communications, 2009.

[14] Rene Mayrhofer and Hans Gellersen, "Shake Well Before Use: Intuitive and Secure Pairing of Mobile Devices", IEEE TRANSACTIONS ON MOBILE COMPUTING, 2009, Vol. 8, No. 6, pp. 792-806

[15] Arun Kumar, Nitesh Saxena, Gene Tsudik, and Ersin Uzun, "A comparative study of secure device pairing methods", Pervasive and Mobile Computing, 2009, Vol. 5, pp. 734-749.