Secure Device-to-Device Communications over WiFi Direct

Wenlong Shen, Bo Yin, Xianghui Cao, Lin X. Cai, and Yu Cheng

Abstract

D2D communications facilitate proximal devices to directly communicate with each other, bypassing cellular base stations or access points, and bring many benefits such as improvement in both spectral efficiency and energy efficiency. Among existing D2D enabling techniques, the recently released WiFi Direct is one promising protocol that offers high data rate D2D communications in local areas. However, WiFi Direct is susceptible to security threats due to the open access of wireless channels and lack of security infrastructures. In this article, we identify several attacks that challenge WiFi-Direct-based D2D communications. Since pairwise key establishment lies in the area of securing D2D communications, we introduce a short authentication-string-based key agreement protocol and analyze its security performance. We also integrate the SAS-based key agreement protocol into the existing WiFi Direct protocol, and implement it using Android smartphones.

Device-to-device (D2D) communications allow spatially proximal devices to directly talk bypassing cellular base stations (BSs) or access points (APs) [1–3]. Such a paradigm shift from two-hop communications to one-hop ones can bring many benefits such as spectral efficiency improvement, energy saving, and delay reduction. Without the need for infrastructure, D2D technology facilitates mobile users sharing instant information (e.g., pictures and videos) with each other even in areas out of cellular coverage or with no AP [4]. It is becoming an important enabling technology for mobile social networks [5] so that friends in the vicinity can be automatically identified and paired up for direct communications. Moreover, it is evolving to enable so-called mobile ad hoc clouds, which exploit untapped resources of a number of proximal devices to provision cloud services such as data and computation offloading.

For devices within small areas (e.g., an office or a home), WiFi is a good option for providing high data rate D2D communications at relatively low cost. Evolved from WiFi, the WiFi Direct specification, recently released by the WiFi Alliance, suits D2D applications by removing the intervention of an AP in both D2D link setup and data communications. Alternatively, it allows devices to dynamically establish a peer-to-peer group and negotiate who takes the role of AP as the group owner (GO) [6]. Devices associated with a GO are called clients. Unlike other D2D enabling techniques in local area networks, such as WiFi in ad hoc mode and IEEE 802.11z, WiFi Direct provisions the same quality of service (QoS) and energy saving mechanisms as infrastructure-based WiFi.

Due to the open access nature of wireless communications, D2D communications are vulnerable to a variety of attacks, which raises critical security challenges [7–9]. For example, adversaries may probe sensitive user data (e.g., the contact list stored in smartphones) by listening to certain D2D links. An attacker may also pretend to be another one to set up D2D links with unsuspecting users. Without a trusted infrastructure such as an AP, it is the D2D users' responsibility to secure their communications and protect their sensitive data from various kinds of attacks. Despite the WiFi Protection Setup (WPS) mechanism inherited from the WiFi specification, WiFi-Direct-based D2D communications cannot be spared from those security challenges. Attacks (e.g., denial-of-service [DoS]) are difficult to prevent with WPS [8]. Moreover, studies have shown that WPS has its own security holes, which can be leveraged by smart adversaries to establish unsafe D2D links. With the proliferation of smartphones and great potential demand for D2D communications, security protection mechanisms are urgently needed.

In the literature, there are few studies on the security aspect of D2D communications. In this article, we discuss the security challenges and identify a number of attacks for D2D communications with WiFi Direct, for example, man-in-the-middle attack and DoS. Pairwise key establishment lies in the kernel for securing D2D links. With the established secure key, a variety of cryptographic encryption algorithms can be implemented to secure D2D communication. To this end, we introduce a short authentication-string-based key agreement protocol. We implement the proposed protocol for file sharing applications in a real system using Android smartphones. Experiments validate the efficiency and effectiveness of the proposed protocol.

The remainder of this article is organized as follows. The following section gives a brief introduction to WiFi Direct. Security challenges are then discussed. Next, we present the proposed key agreement protocol and the experiment results. The following section discusses some future research issues. Finally, we give concluding remarks.

WiFi Direct Overview

The WiFi Direct protocol enables two devices to establish a D2D connection without the help of APs. Figure 1 shows the procedure for D2D connection establishment using WiFi Direct. First, two devices perform channel probing and discover each other. Then they negotiate to determine the group owner (GO), which operates as an AP for this D2D connec-

Wenlong Shen, Bo Yin, Lin X. Cai and Yu Cheng are with Illinois Institute of Technology.

Xianghui Cao is with Southeast University.

0890-8044/16/$25.00 © 2016 IEEE

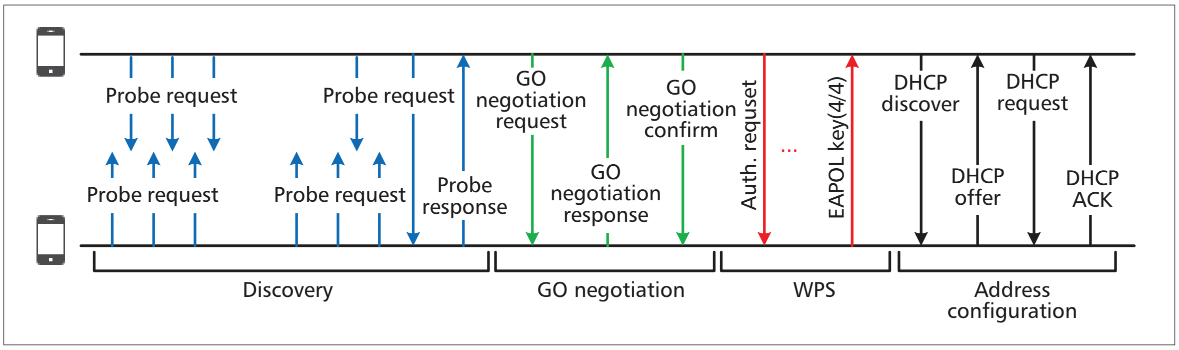


Figure 1. WiFi Direct protocol.

tion in a voluntary or random manner. During the handshake process, each device sends a GO intent value, and the device with the highest value becomes the GO. After the devices have agreed on their respective roles, the GO initiates the WiFi security setup using WPS, and conducts a Dynamic Host Configuration Protocol (DHCP) exchange to set up the IP addresses for both devices. Thus, the D2D connection between these two devices is established.

WiFi Direct is built on the IEEE 802.11 infrastructure mode, and thus can be seamlessly implemented by legacy WiFi devices. Besides, WiFi Direct inherits the features of traditional WiFi, including QoS, power saving, and security mechanisms, and has the capability of forming a more stable and secure D2D underlying network than traditional ad hoc networks. Two typical WiFi Direct application scenarios are illustrated in Fig. 2:

- Two (or multiple) devices form a local ad hoc network. Messages and files can be shared among this network at no extra cost. This is particularly useful when there is no cellular connection or AP available.
- By WiFi Direct, a cellular-enabled device can share its Internet connection with other devices.

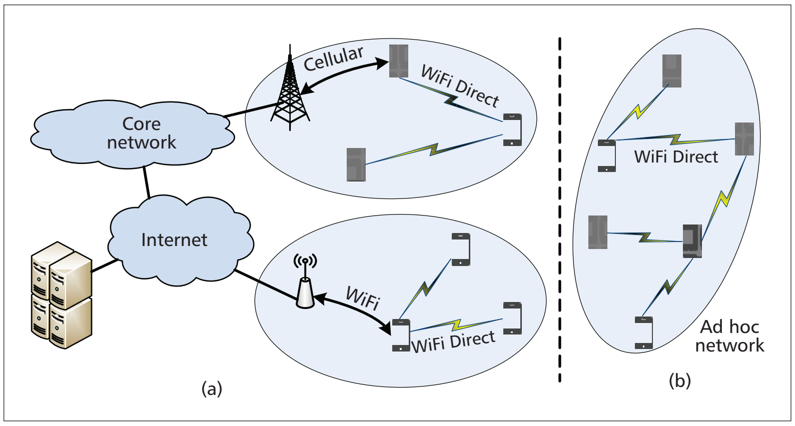


Figure 2. WiFi Direct application scenarios: a) one device shares its cellular connection; b) multiple devices form an ad hoc network.

Security Challenges for D2D Communication via WiFi Direct

Despite all the benefits of the emerging WiFi Direct protocol, security is one of the major concerns that need to be well addressed before this technique is widely accepted and implemented. However, due to the open access and broadcast nature of wireless channels, WiFi Direct is threatened by a variety of attacks.

Attack Modes

As shown in Fig. 3, the most common attack vectors include surreptitious eavesdropping, message modification, and node impersonation.

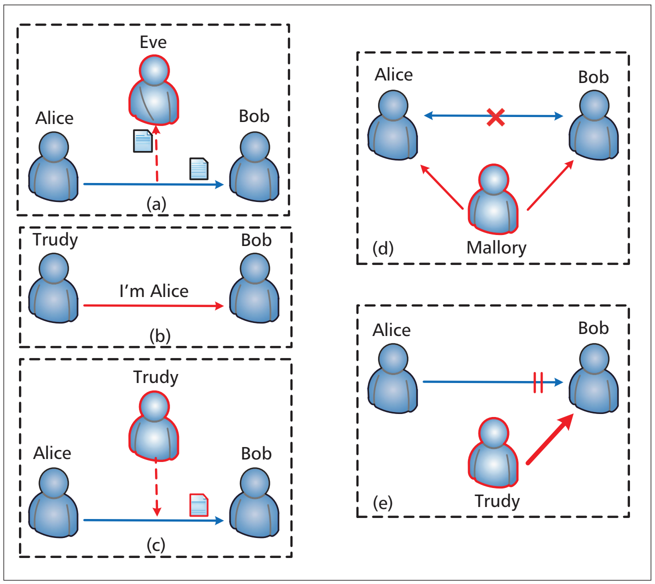


Figure 3. Attack modes: a) eavesdropping; b) impersonation; c) message modification; d) MITMA; e) DoS attack.

Eavesdropping: The open access nature of the wireless channel offers the attackers the convenience to listen to any ongoing traffic over the wireless channel. Eavesdropping is easily achievable by any traffic sniffing software, even without requiring the attacker to have any advanced security or computer technology. If the network traffic data is not encrypted, privacy information involved in the D2D communications, including personal data and location information, is exposed to the eavesdropper.

Impersonation: The attacker may impersonate a legitimate user by transmitting a message with the legitimate user's medium access control (MAC) and IP addresses. Without authentication information, the receiver has no way to tell the real identity of the transmitter. A well educated attacker can launch such an attack by programming its network adapter. This kind of attack is highly achievable.

Message Modification: To modify ongoing traffic without being noticed by legitimate users is much more difficult but still possible. By using advanced full duplex radio techniques, the attacker is able to receive and transmit signals simultaneously. When the attacker captures ongoing traffic between the transmitter and the receiver, he/she immediately impersonates the transmitter, modifies the message payload as his/her wish, and sends the modified message to the receiver using a directional antenna at significantly higher power, producing a capture effect. In such a case, the receiver will only decode the modified message, while the transmitter has no idea that there is an impersonation going on due to the directional antenna the attacker uses.

Apart from the aforementioned three basic attack modes, WiFi Direct communication faces some other, more sophisticated potential security threats. These security threats either take advantage of the protocol defects or are combinations of the basic attack modes.

Man-in-the-Middle Attack: MIMTA is a well-known attack in wireless communications, in which the attacker makes independent connections with legitimate users, and relays and modifies the messages between them to make them believe they are talking directly to each other over a private connection. However, the entire communication is under the attacker's control. To avoid detection by legitimate users, the attacker needs to intercept the original communication messages and forge new ones, similar to the above message modification and impersonating attacks. To defend against MIMTA, mutual authentication is required.

Denial of Service Attack: A typical attack model is that in the discovery phase of the WiFi Direct protocol, the attacker sends bogus requests and responses to the legitimate users to launch a flooding attack by saturating the target device with discovery requests so that it cannot respond to legitimate requests. Another possible DoS attack model is in the application scenario in Fig. 2a, where one of the devices may launch such an attack by uploading heavy data to the Internet via the WiFi Direct GO. Because the WiFi Direct underlying network operates at a much higher bandwidth than the GO's cellular uplink connection, the data uploading traffic has little impact on the underlying WiFi Direct network, but consumes all of the GO's uplink bandwidth. Thus, the remaining devices are not able to properly utilize the GO's Internet connection [8].

The prevention of DoS attacks requires countermeasures in the design of peer discovery protocols and misbehavior flow detection mechanisms. However, other listed attack modes are ready to be solved by cryptography methods: encrypting the network traffic can protect data privacy from eavesdroppers; a message authentication code (MAC) or digital signature can guarantee the integrity and authentication of the communication messages. Symmetric and asymmetric key cryptography algorithms have been well studied; each of them can provide desired protections to the network data, so the essential problem becomes how to establish or distribute the secure key between the users of WiFi Direct.

Limitations of WiFi WPS

WiFi Direct implements WiFi Protected Setup (WPS) to establish a secure connection via a PIN or pushbutton configuration (PBC). The design goal of WPS is to allow home users with little wireless security knowledge to easily set up a secure WiFi connection. In WPS, the WiFi Direct GO works as the registrar, which issues network security credentials to the enrollee (WiFi Direct client). Following WPS, WiFi Protected Access (WPA) or WiFi Protected Access II (WPA2) protocol generates secure keys and provides protection to the network traffic. However, a major security flaw of the PIN feature of WPS has been discovered[10], which allows the attacker to recover the WPS PIN within a short time. With the WPS PIN,

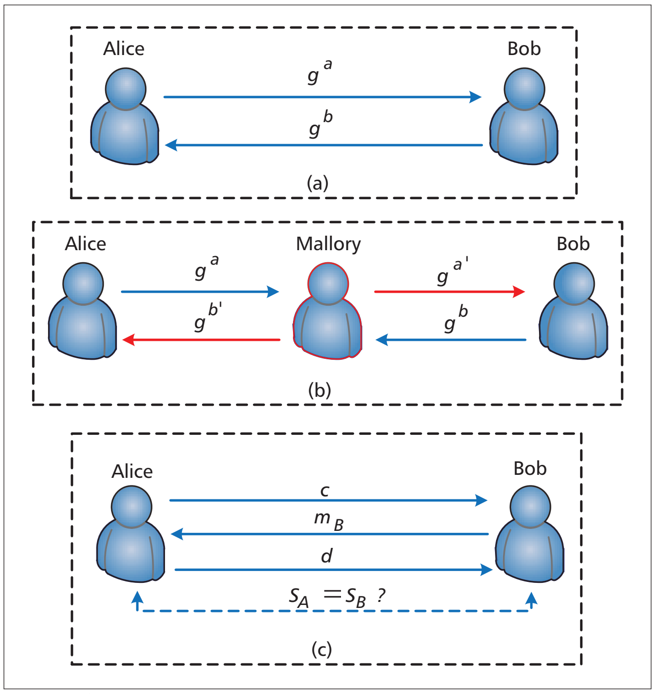


Figure 4. Pairwise key establishment: a) Diffie-Hellman key agreement protocol; b) the MITMA in Diffie-Hellman; c) the SAS-based key agreement protocol.

the secret key generated by WPA/WPA2 can also be revealed. Besides, the PBC feature of WPS is also vulnerable[11]: an active attacker can gain access to the registrar or enrollee, or intercept and modify any messages between them.

Pairwise Key Establishment

To secure WiFi Direct D2D communications, a cryptography key is required to perform the cryptography algorithms such as encryption and authentication. However, how to generate and distribute the cryptography keys among WiFi Direct users is not a simple task. In this section, we analyze the problems of key generation without a prior shared secret, and briefly introduce one typical device pairing protocol utilizing a short authentication string (SAS).

A key distribution center (KDC) or a public key infrastructure (PKI) would be the first choice for key distribution and management in most cases. However, in the WiFi Direct application scenario, in most cases users establish their D2D links in a dynamic and distributed way; hence, security infrastructures and certification authorities may not exist. On the other hand, due to the diversity of device manufacturers and lack of unified standards, preloading secure keys into devices is neither efficient nor practical. Thus, establishing a session key during the WiFi Direct group formation process would be the proper choice for WiFi Direct D2D communications.

Security Concerns for Key Establishment

To let two end users of the D2D link interactively negotiate a secret key would be one straightforward method, but such a secret key established by human negotiation is in most cases too weak. An attacker with significant computation power can easily crack this weak secret key by brute force within a short time. In research and real world applications, the Diffie-Hellman key agreement protocol is the most common approach for two individuals to establish a shared secret key.

As shown in Fig. 4a, Diffie-Hellman key agreement allows two individuals with no prior knowledge to jointly establish a shared secret key, even though they can only exchange messages over public insecure communications channels. The procedure of Diffie-Hellman key agreement is as follows: Assume g is publicly known to Alice and Bob (if not, Alice can broadcast it), Alice and Bob randomly generate a and b, then compute g^a and g^b, respectively. Alice sends g^a to Bob and Bob sends g^b to Alice. At the final stage, Alice computes $K = (g^a)^b$, and Bob computes $K = (g^a)^b$. Both Alice and Bob will arrive at the same value, since $(g^a)^b$ and $(g^a)^b$ are equal. K will be the shared secret between Alice and Bob, and subsequently can be used for encryption or authentication in their future communications. The implementation of Diffie-Hellman key agreement requires some amount of computation capacity, since a and b can be large numbers. However, mainstream mobile devices on today's market have achieved gigahertz processor frequency; thus, using Diffie-Hellman key agreement to establish a secure enough shared secret, say, 156 bits, can be conducted within seconds.

It is well known that the Diffie-Hellman key agreement protocol is vulnerable to the MITMA. As illustrated in Fig. 4b, since g^a and g^b are exchanged over the public channel, there is no way for Alice to know for sure whether g^b comes from Bob, and vice versa. Alice will establish a shared secret with whoever transmits g^b, who might not be Bob. The essential reason that the MITMA is possible lies in the lack of mutual authentication between these two individuals. It has been formally proved using BAN logic that device authentication using a single channel is not possible [12]. To provide the desired authentication, an out-of-band channel is desired. Such a channel, sometimes called a "human assisted channel," cannot be manipulated by the attackers, and thus can be trusted. With an out-of-band channel, one intuitive solution for Alice and Bob to authenticate each other would be that they both compute a hash value of the obtained shared key and then compare the hash values $h(K)$ via the out-of-band channel (e.g., visually or verbally). If the mutual authentication process agrees, both Alice and Bob can confirm that the result of the Diffie-Hellman key exchange is valid between them.

The main issue about the above authentication procedure using hash is that the number of bits that need to be compared is too large. The output of a cryptographic hash function is usually over 128 bits (32 hexadecimal digits), so visually or verbally comparing them is inefficient and fallible. Using truncation of the hash code can reduce the number of bits needed to be compared and hence improve user friendliness, but at the same time drastically reduce the security level. For example, [13] shows that an attacker with significant computing power can crack a 32-bit truncated hash code in less than 1 s.

A Short-Authentication-String-Based Key Agreement Protocol

Much research attention has been focused on designing a pairwise key agreement protocol that involves minimal mutual authentication or human interaction. Here we briefly introduce one key agreement protocol that is based on the authentication of a very short string. This protocol provides optimal security level in regard to the length of bits required to be mutually authenticated.

As shown in Fig. 4c, the short authentication string (SAS)-based key agreement protocol utilizes a cryptography commitment scheme. A commitment scheme enables one to hide a chosen value through a commit operation, transforming this value into a commitment/opening pair. Anyone who obtains this pair of values is able to reveal the committed value deterministically via an open operation. The commitment value alone, however, leaks no information about the hidden value. An efficient construction of a commitment scheme can be achieved by using a cryptographic hash function [14]. Specifically, assuming that H is a cryptographic hash function, we have the following commitment scheme:

- **Commit**: Given x, randomly pick $r \leftarrow \{0, 1\}^n$ and compute c $= H(x, r)$.
- **Open**: Let $d = (x, r)$. Output x if $c = H(x, r)$.

The efficiency and security level of such a commitment scheme directly depends on the hash function in use.

By adopting a commitment scheme for mutual authentication, the SAS-based key agreement protocol enhances the security level of the traditional Diffie-Hellman key agreement protocol, protecting it from an MITM attack. Besides their Diffie-Hellman parameters g^a and g^b, Alice and Bob need to generate a value in the form of the concatenation of their respective identifier, Diffie-Hellman parameter, and a k-bit random string. The chosen value of Alice is denoted as $m_A = \mathrm{ID}_A \| g^a \| \mathrm{N}_A$. Then Alice employs a commitment scheme to commit m_A. (c, d) is the commitment/opening pair.

In order to establish a secret key, Alice and Bob exchange a series of messages over the public channel. To begin with, Alice sends the commitment value c to Bob. Bob then sends his value m_B to Alice. After receiving m_B, Alice sends the opening value d of m_A from which Bob can reveal the chosen value of Alice through the open operation.

Before generating a shared secret key based on the exchanged Diffie-Hellman parameters, Alice and Bob need to authenticate each other over an out-of-band channel. In SAS-based key agreement, Alice and Bob generate a k-bit authentication string $S_A = \mathrm{N}_A \oplus \mathrm{N}_B$ and $S_B = \mathrm{N}_A \oplus \mathrm{N}_B$ where N_B and N_A are extracted from m_B or the revealed m_A, respectively. Then, over the trusted channel, they verify whether $S_A = S_B$. The mismatch of these strings indicates the existence of an MITM attack. Alice and Bob would stop the current key establishment process.

Security Analysis

In this SAS-based key agreement protocol, Alice has to commit on m_A before actually seeing m_B, and Bob has to submit an m_B before actually seeing m_A. Thus, if Eve attempts to launch an MITMA, no matter what attacking strategy Eve applies, he has to first commit to or submit an m_E, which contains his authentication string N_E. Suppose Eve initiates a protocol with Bob pretending to be Alice; first, she will commit to an $m_E = \mathrm{ID}_A \| g^e \| \mathrm{N}_E$ and send the commit value c_E to Bob. After receiving Bob's reply m_B, Eve can modify m_B into $m'_B = \mathrm{ID}_B \| g^e \| \mathrm{N}_B$ and forward it to Alice. But when it comes to the mutual authentication stage, Alice and Bob will compare $S_B = \mathrm{N}_B \oplus \mathrm{N}_E$ with $S_A = \mathrm{N}_A \oplus \mathrm{N}_B$ through the out-of-band channel and find out that $S_A \neq S_B$. The only chance that Eve can launch a successful attack by making Alice and Bob agree on the authentication string is to have $\mathrm{N}_E = \mathrm{N}_A$, and she can only achieve this by random guessing the right N_A, with probability 2^{-k}. If Eve attempts to launch an attack by replying $M_E = \mathrm{ID}_B \| g^e \| \mathrm{N}_E$ to the protocol initiator Alice, similar analysis follows.

The bit length k of the authentication string can be tuned to balance the security level and usability. With a larger k, attackers have a smaller possibility to launch a successful attack, but the two users need to compare a longer string via the out-of-band channel. Usually a 20-bit (5 hexadecimal digits) authentication string is secure enough, which provides a security level similar to an ATM.

A Case Study

We implement a secure file transfer Android application by incorporating secure key establishment functionality into a conventional WiFi Direct application. Our application allows users to establish pairwise secret keys which can then be used to encrypt data transmitted via D2D connection. The modified

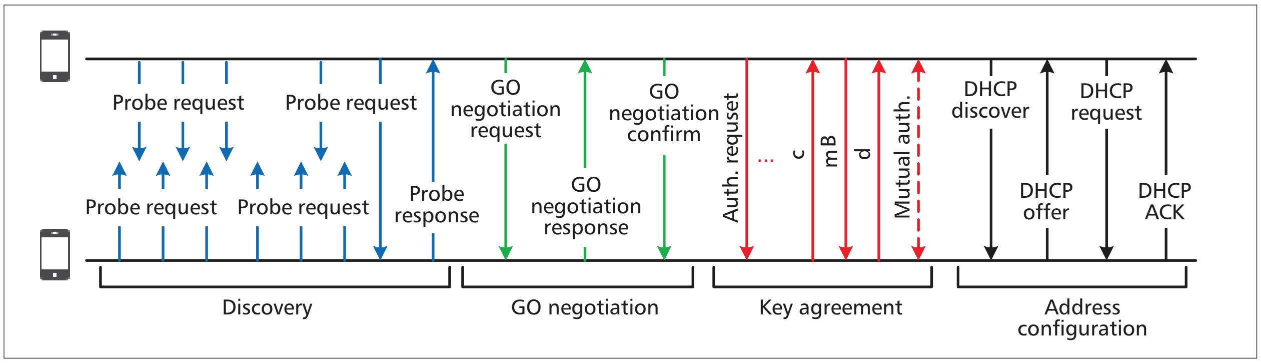


Figure 5. Secure WiFi Direct protocol.

WiFi Direct protocol is shown in Fig. 5. Screenshots of our developed application are illustrated in Fig. 6. Our application, based on the WiFi Direct Demo [15], is implemented on two Google Nexus 5 smartphones with Android 4.4 Kitkat. A hash-based commitment scheme [14] is employed in our implementation, which would not incur too much computational burden during the key agreement phase.

We list, in hexadecimal values, some of the Diffie-Hellman parameters and authentication for *N* strings of one instantiation. The prime number *p* is set to be 40 digits, which could generate a roughly 130-bit secret key. The authentication string consists of 5 hex characters, which is easy to compare and can achieve a strong security level. The overall running time for the key agreement process, including the computation time as well as the communication delay, is negligible on Nexus 5 with its 2.26 GHz processor.

Future Research Issues

Out-of-Band Secure Channels

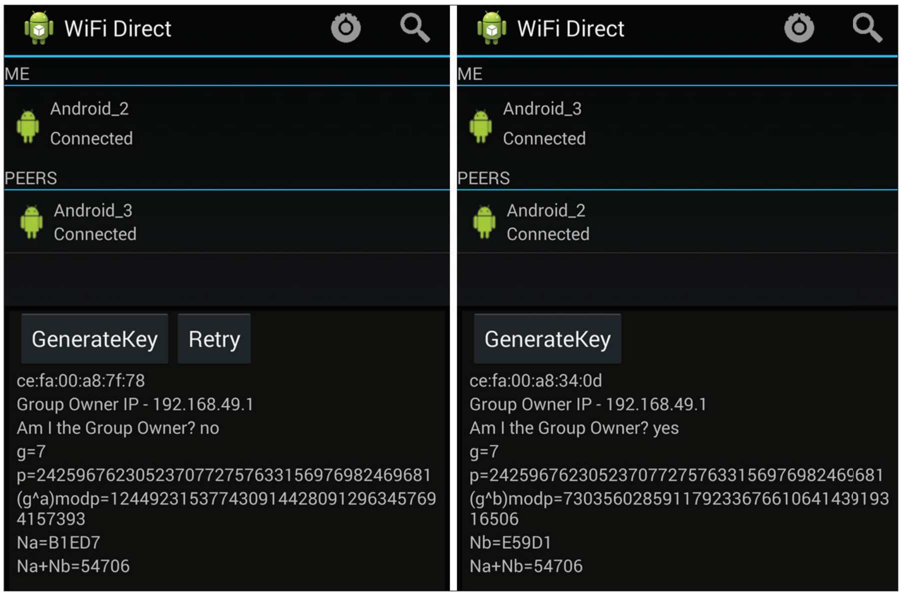
The introduced SAS-based key agreement protocol requires users to compare an authentication string using an out-of-band channel. Displaying the authentication string on a user device's screen is one straightforward way; however, features of recent smartphones allow users to compare the authentication string in more convenient and efficient ways. For example, the authentication string can be displayed in the form of quick response (QR) codes, and the users can simply verify the QR codes using smartphone cameras. Some smartphone models feature a near field communication (NFC) module, which can also be used as an out-of-band channel for the mutual authentication process. How to explore and utilize the advanced features of today's smartphones as the out-of-band channel in the key agreement protocols to reduce the authentication overhead and enhance user experience is one important research issue.

Physical Layer Key Generation

Physical-layer-based secret key generation methods have been proposed in recent years as alternative solutions for the traditional Diffie-Hellman key agreement protocol. Unlike Diffie-Hellman key agreement, with security relying on the computational hardness of the discrete logarithm problem, the physical-layer-based method is based on the randomness and uniqueness of wireless fading channel properties: temporal variation, spatial variation, and reciprocity. Physical-based key generation can achieve information-theoretical secrecy and provide more flexibility for securing wireless links. However, the key generation rate in current physical layer methods is very low. Users have to send lots of channel probing packets to achieve a secret key with enough bits and randomness. More research attention is needed to reduce the communication overhead and key generation time.

Conclusion

In this article, we have proposed a secure WiFi Direct protocol that can be used for secure D2D communications. We have first analyzed the potential security threats and challenges for the emerging WiFi Direct protocol. Then we have discussed how to efficiently establish a cryptography key to protect the WiFi Direct communications. We have also demonstrated an experiment implementation of the introduced key agreement protocol on Android smartphones. Finally, future research issues have been discussed.



```
Android_2
Connected
Android_3
Connected
ce:fa:00:a8:7f:78
Group Owner IP - 192.168.49.1
Am I the Group Owner? no
g=7
p=2425967623052370772757633156976982469681
(g^a)modp=12449231537743091442809129634576
94157393
Na=B1ED7
Na+Nb=54706
```

```
Android_3
Connected
Android_2
Connected
ce:fa:00:a8:34:0d
Group Owner IP - 192.168.49.1
Am I the Group Owner? yes
g=7
p=2425967623052370772757633156976982469681
(g^b)modp=73035602859117923367661064143919
316506
Nb=E59D1
Na+Nb=54706
```

Figure 6. Experiment on Android smartphones.

Acknowledgments

This work was supported in part by the NSF under grants CNS-1117687 and CNS-1320736, and the National Natural Science Foundation of China under grants 61203036 and 61573103.

References

[1] K. Doppler *et al.*, "Device-to-Device Communication as an Underlay to LTE-Advanced Networks," *IEEE Commun. Mag.*, vol. 47, no. 12, 2009, pp. 42–49.
[2] J. Liu *et al.*, "Device-to-Device Communications Achieve Efficient Load Balancing in LTE-Advanced Networks," *IEEE Wireless Commun.*, vol. 21, no. 2, 2014, pp. 57–65.
[3] L. Wang *et al.*, "Sociality-Aware Resource Allocation for Device-to-Device Communications in Cellular Networks," *IET Commun.*, vol. 9, no. 3, 2015, pp. 342–49.

[4] P. Phunchongharn, E. Hossain, and D. I. Kim, "Resource Allocation for Device-to-Device Communications Underlying LTE-Advanced Networks," *IEEE Wireless Commun.*, vol. 20, no. 4, Aug. 2013, pp. 91–100.
[5] M. Dong *et al.*, "Quality-of-Experience (QoE) in Emerging Mobile Social Networks," *IEICE Trans. Info. Sys.*, vol. 97, no. 10, 2014, pp. 2606–12.
[6] D. Camps-Mur, A. Garcia-Saavedra, and P. Serrano, "Device-to-Device Communications with WiFi Direct: Overview and Experimentation," *IEEE Wireless Commun.*, vol. 20, no. 3, 2013.
[7] W. Shen *et al.*, "Secure Key Establishment for Device-to-Device Communications," *Proc. IEEE GLOBECOM*, 2014.
[8] A. Hadiks *et al.*, "A Study of Stealthy Denial-of-Service Attacks in WiFi Direct Device-to-Device Networks," *Proc. Annual IEEE Consumer Commun. and Networking Conf.*, work in progress, Jan. 2014.
[9] H. Kwon *et al.*, "Secure Device-to-Device Authentication in Mobile Multi-Hop Networks," *Wireless Algorithms, Systems, and Applications*, 2014, pp. 267–78.
[10] S. Viehbock, "Brute Forcing WiFi Protected Setup," *WiFi Protected Setup*, 2011.
[11] S. Gollakota *et al.*, "Secure in-Band Wireless Pairing," *Proc. USENIX Security Symp.*, 2011.
[12] W. Claycomb and D. Shin, "Extending Formal Analysis of Mobile Device Authentication," *J. Internet Services and Info. Security*, vol. 1, no. 1, 2011, pp. 86–102.
[13] C. Gehrmann, C.J. Mitchell, and K. Nyberg, "Manual Authentication for Wireless Devices," *RSA Cryptobytes*, vol. 7, no. 1, 2004, pp. 29–37.
[14] R. Pass, "On Deniability in the Common Reference String and Random Oracle Model," *Advances in Cryptology 2003*, 2003, pp. 316–37.
[15] WiFi Direct Demo; http://www.androidside.com/docs/resources/samples/WiFiDirectDemo/index.html

Biographies

WENLONG SHEN (wshen7@hawk.iit.edu) received his B.E. degree in electrical engineering from Beihang University, Beijing, China, in 2010, and his M.S. degree in telecommunication from the University of Maryland, College Park, in 2012. He is currently pursuing his Ph.D. degree with the Department of Electrical and Computer Engineering, Illinois Institute of Technology, Chicago. His current research interests include network security, mobile cloud computing, and D2D communications.

BO YIN (byin@hawk.iit.edu) received his B.E. degree in electronic information engineering and M.E. degree in electronic science and technology from Beihang University in 2010 and 2013, respectively. Currently, he is a Ph.D. candidate in the Department of Electrical and Computer Engineering, Illinois Institute of Technology. His research interests include mobile cloud computing and network security.

XIANGHUI CAO (xhcao@seu.edu.cn) received his B.S. and Ph.D. degrees in control science and engineering from Zhejiang University, Hangzhou, China, in 2006 and 2011, respectively. From July 2012 to July 2015, he was a senior research associate with the Department of Electrical and Computer Engineering, Illinois Institute of Technology. Currently he is an associate professor with the School of Automation, Southeast University, Nanjing, China. His research interests include cyber physical systems, wireless networked control systems, wireless network protocols and network security. He served as Publicity Co-Chair for ACM MobiHoc 2015, Symposium Co-Chair for IEEE/CIC ICCC 2015, and TPC member for many conferences. He also serves as an Associate Editor for *KSII Transactions on Internet and Information Systems*, *Security and Communication Networks* (Wiley), and the *International Journal of Ad Hoc and Ubiquitous Computing*. He was a recipient of the Best Paper Runner-Up Award from ACM MobiHoc 2014.

LIN X. CAI (lincai@iit.edu) received her M.A.Sc. and Ph.D. degrees in electrical and computer engineering from the University of Waterloo, Canada, in 2005 and 2010, respectively. She was a postdoctoral research fellow in the Electrical Engineering Department at Princeton University in 2011. She worked as a senior engineer at Huawei US Wireless R&D Center before she became an assistant professor with the the Department of Electrical and Computer Engineering, Illinois Institute of Technology in August 2014. Her research interests include green communication and networking, broadband multimedia services, radio resource and mobility management, and cognitive radio networks.

YU CHENG (cheng@iit.edu) received B.E. and M.E. degrees in electronic engineering from Tsinghua University in 1995 and 1998, respectively, and a Ph.D. degree in electrical and computer engineering from the University of Waterloo in 2003. He is now an associate professor in the Department of Electrical and Computer Engineering, Illinois Institute of Technology. His research interests include wireless network performance analysis, network security, and next-generation Internet technology. He received Best Paper Awards at QShine 2007 and IEEE ICC 2011, and a Runner-Up Best Paper Award at ACM MobiHoc 2014. He received the National Science Foundation (NSF) CAREER Award in 2011 and IIT Sigma Xi Research Award in the junior faculty division in 2013. He has served as Symposium Co-Chair for IEEE ICC and IEEE GLOBECOM, and Technical Program Committee CoChair for WASA 2011 and ICNC 2015. He is a founding Vice Chair of the IEEE ComSoc Technical Subcommittee on Green Communications and Computing. He is an Associate Editor for *IEEE Transactions on Vehicular Technology* and the New Books & Multimedia Column Editor for *IEEE Network*.