Secure Audio Pairing

Brad Girardeau BGIRARDE@CS.STANFORD.EDU
Dan Boneh DABO@CS.STANFORD.EDU
Stanford University, Stanford, CA 94305 USA

Abstract

Securely pairing previously unknown devices without a trusted third party is a challenging, yet increasingly common problem. Most existing approaches rely on the user to complete a verification task for security that is easily skipped, or they impose too great a burden on the user, limiting applications that require frequent pairing of devices. We present a method that uses audio to streamline user interaction and improve usability. Since users can easily hear if a different device is playing, audio forms an implicit authenticated channel. This allows for automated authentication of a key exchange. We build on previous work to improve the performance and reliability of the audio channel in practice and apply this method to securing Bluetooth Low Energy connections without active user involvement.

1. Introduction

From sensors to smart cars, innovative applications connect devices together. Yet even as technology increasingly communicates, user data must remain protected. When devices communicate wirelessly, attackers have several opportunities to intercept sensitive data. They can passively listen, or they can actively insert themselves as a man in the middle (MITM) to intercept sensitive data. Protecting against these attacks without significantly harming the user experience is a key component of modern connected application design.

The core technical challenge is *pairing* previously unknown devices to establish a secure connection. A secure device pairing sets up a communications channel that is both confidential and authenticated; the devices know they are communicating with each other while hiding information from other listeners. If the devices share a secret key, many standard authenticated encryption systems can be used to create such a channel. In addition, generating a shared secret key over a public connection can be accomplished using existing algorithms like the Diffie-Hellman key exchange (3). These building blocks bootstrap a confidential channel over a public connection, but they are unauthenticated, still vulnerable to MITM attacks. An additional authentication component is needed during key exchange to create a secure connection.

Authenticating messages over an insecure channel is a difficult task. Many approaches involve a certificate authority infrastructure or a trusted third party, like Kerberos (4), but these are impractical for pairing within a large, diverse device ecosystem. Instead different "out-of-band" (OOB) channels are used that can naturally authenticate messages between participants, even if the messages are not necessarily kept confidential. For example, two users can communicate face to face and be assured that they are speaking to the correct person, although someone may still be eavesdropping on the conversation. A variety of OOB channels with different security properties have been proposed for use in secure device pairing.

In this paper, we present a system that uses an audio OOB channel to securely pair devices. This leverages and extends existing techniques to increase usability by decreasing user interaction. Usability directly impacts security in the device pairing context by improving the user's willingness and ability to correctly authenticate messages. Audio is particularly well suited for this purpose. It demonstrates that the intended device is pairing by simply having a user locate which device is making sound. Localizing sounds (and noticing if extra devices are making sound) is an accurate, almost subconscious process for humans (4). Audio hardware is also widespread; many devices, like phones, have speakers and microphones. This system

can be widely deployed to streamline the pairing process by minimizing user interaction while maintaining security.

2. Related Work

Significant research has focused on methods of secure device pairing. Many systems are based on using short authenticated strings (SAS) over an OOB channel, introduced by Vaudenay in 2005 (11). Two common approaches involve the user directly authenticating information for an OOB channel by entering a PIN code or verifying two short strings are equal, both of which are seen in the Bluetooth Low Energy specification (9). While these methods are effective and easy to understand, they involve significant user interaction: frequently entering a PIN is annoying and verifying that strings are equal is easily (and likely to be) skipped.

To address these issues, a variety of systems with different OOB channels have been proposed. *Seeing Is Believing* uses a camera phone as a visual channel (8), and *Shake Well Before Use* describes using accelerometer data as an authenticated channel (7). Using audio specifically as an OOB channel has also been explored. The *Loud and Clear* system encodes data as syntactically valid English sentences, which the user then listens to and compares (5). *HAPADEP* extends this to also include comparing melodies (10). Several surveys of pairing methods like (2) provide further discussion.

While these systems are promising, they still require active user interaction. This weakens usability, and often by extension security as users may not complete the verification task correctly. Instead we propose using audible sound's inherent MITM resistance to enable secure device pairing by passive users. One recent paper from 2014 also uses this idea in *MVSec* to pair phones and cars (6). In this paper, we go beyond that work by analysing and improving the audio channel, as well as discussing practical solutions for creating a cross-platform pairing system.

3. Protocol

The pairing system consists of a key exchange protocol that is built on two communications channels, one authenticated and one unauthenticated. The protocol integrates a SAS protocol with a standard Diffie-Hellman key exchange to authenticate the public key parameters. This essentially follows the DH-SC protocol presented in (11) and (1).

3.1. Steps

We present the protocol briefly here; it is presented and analyzed extensively in the work referenced above.

The basic idea in this protocol is to do a standard key exchange, with an additional verification value generated at the end that must be the same on each side to ensure an attacker did not manipulate any earlier messages. To create this verification message, each device commits to a random string that is not known to the attacker and sends that commitment to the other device early in the protocol. Then to successfully impersonate the other device, the attacker will have to guess the random string, or else they will commit to the wrong value. At the end, the authenticated channel is used to check that a verification value based on the random strings matches for both devices.

1. Each device chooses a random k-bit string and also constructs a Diffie-Hellman key-exchange message. Note that an attacker has a 2^{-k} chance of success (11), and a typical value for k is 20 (4).

2. Each device generates a commitment to the message that when opened gives the random string.

3. Each device sends its message and commitment (but not random string) to the other over the unauthenticated channel.

4. Then each device sends the opening value for its commitments to the other over the unauthenticated channel.

5. Now each device can recover the other's random string by opening the commitment it was sent (assuming an attacker has not modified messages).

6. The short authenticated string (SAS) is the XOR operation applied to the two random strings. Each device sends each other the SAS it computed over the authenticated channel. If the SASs match, the protocol and pairing are successful.

In this standard protocol, both sides send and verify the SAS at the end. This is needed for security in a generic context, but in many cases, only one side needs to send the full authenticated string over the audio channel (while the other receives it and verifies). This is because there is typically already some sort of notification to the user from the receiving device that pairing succeeded, which is effectively an authenticated message that replaces the need for another in this protocol.

4. Audio Channel

The audio channel is the authenticated channel in this protocol. It ensures that the devices being paired are the desired ones by verifying that they share an authenticated string. As the user-facing part of the protocol, it requires the most design consideration. Users will listen to the audio component each time they pair a device, and part of the security model assumes that they will notice if their device is not the only one emitting sound (which would indicate a MITM attack). This implies the sound must be noticeable, and relatively pleasing so that users do not mind its role in the pairing process. In addition, the sound should be robust to some degree of background noise.

To accomplish these goals, the sender encodes data by mapping 3 bits to 8 notes. These notes are played using a MIDI synthesizer on one of a selection of instruments. This creates a random sequence that is musical, catching a user's ear. To encode 20 bits, which is a standard length for the Vaudenay SAS protocol, seven short notes are needed. This means the audio portion of the protocol is relatively fast, taking under two seconds, compared to 5-6 seconds in the MVSec scheme (6).

The receiver then processes this audio in real-time to detect whether the data it expected was sent. Notably the receiver does not need to recognize unknown data, only needing to determine whether it and the sender share a short string. This framing of the problem facilitates a robust and extensible detection pipeline.

4.1. Detection Pipeline

The audio detection system is responsible for deciding whether an input audio signal contains a known sequence of notes. The idea is to use audio cross-correlation between note signals and the input signal. This calculates similarity data between consecutive segments of the input signal and the desired notes. A decision algorithm then analyzes the similarity data to indicate whether the signal was detected.

When processing live input, new blocks of data arrive at a constant rate and are added to a buffer at least as large as the expected signal. Then at every n'th new offset into the buffer, the detection function is run. n is chosen as small as possible to allow processing the entire new block before the next arrives.

The detection function splits the input data into chunks with the duration of a single note. Each chunk is then correlated with each of the 8 possible notes to produce 8 correlation coefficients per chunk. Then a note decider function runs on each chunk and returns a matching score based on the correlation coefficients and expected value. Finally an aggregator function combines the matching scores to decide if the input signal matched the expected signal well enough.

Both the note decider and aggregator functions are customizable. They currently use a simple threshold and average comparison approach. Future experimentation will evaluate the effectiveness of different functions, possibly using machine learning techniques.

4.2. Effectiveness

Current qualitative results suggest the detection pipeline performs more reliably than other pitch detection methods. The system is robust to low levels of background noise. Talking and music still present a challenge, as they produce noise at similar frequencies to the desired signal. However the detection pipeline should continue to improve significantly as the note decider and aggregator functions become more sophisticated.

5. Bluetooth Channel

It is possible to use any near-field communication technology as the unauthenticated channel. In this project, Bluetooth Low Energy (BLE) is chosen for its wide adoption in modern connected devices. BLE enables communication between a "central" and "peripheral." While a device can support both roles, the initiating device of the pairing takes on the "central" role and its counterpart the "peripheral." The BLE protocol is not designed for high data transfer rates, limiting writes to 20 bytes at a time. This limitation can be overcome during the key exchange by breaking key messages into multiple writes. Still for some applications, regular Bluetooth is more appropriate. In any case, the audio channel pairing described above remains effective.

6. Cross Platform Implementation

The system implemented is designed to work on a variety of devices. Cross-platform web technology, specifically HTML, CSS, and Javascript, is used to implement all core components and interfaces. On mobile devices, Phonegap provides access to the necessary native APIs on Android and iOS. On desktop operating systems, NodeJS and NW.js give web applications access to native capabilities, in particular Bluetooth access. These cross-platform technologies enable wider adoption of this pairing technology.

7. Conclusion

The system presented here securely pairs devices with minimal, passive user interaction. It does so in a seamless, intuitive way through the use of pleasing musical tones. Furthermore, this solution can be deployed on a wide variety of platforms and devices. It brings an improved experience to the secure pairing process, enabling previously impractical applications for connected devices.

References

[1] M. Cagalj, S. Capkun, and J.-P. Hubaux. Key agreement in peer-to-peer wireless networks. *Proceedings of the IEEE*, 94(2):467–478, Feb 2006.

[2] Ming Ki Chong, Rene Mayrhofer, and Hans Gellersen. A survey of user interaction for spontaneous device association. *ACM Comput. Surv.*, 47(1):8:1–8:40, May 2014.

[3] W. Diffie and M.E. Hellman. New directions in cryptography. *Information Theory, IEEE Transactions on*, 22(6):644–654, Nov 1976.

[4] Michael T Goodrich, Michael Sirivianos, John Solis, Claudio Soriente, Gene Tsudik, and Ersin Uzun. Using audio in secure device pairing. *International Journal of Security and Networks*, 4(1-2):57–68, 2009.

[5] M.T. Goodrich, M. Sirivianos, J. Solis, G. Tsudik, and E. Uzun. Loud and clear: Human-verifiable authentication based on audio. In *Distributed Computing Systems, 2006. ICDCS 2006. 26th IEEE International Conference on*, pages 10–10, 2006.

[6] Jun Han, Yue-Hsun Lin, Adrian Perrig, and Fan Bai. Short paper: Mvsec: Secure and easy-to-use pairing of mobile devices with vehicles. In *Proceedings of the 2014 ACM Conference on Security and Privacy in Wireless Mobile Networks*, WiSec '14, pages 51–56, New York, NY, USA, 2014. ACM.

[7] Rene Mayrhofer and Hans Gellersen. Shake well before use: Authentication based on accelerometer data. In *Pervasive computing*, pages 144–161. Springer, 2007.

[8] J.M. McCune, A. Perrig, and M.K. Reiter. Seeing-is-believing: using camera phones for human-verifiable authentication. In *Security and Privacy, 2005 IEEE Symposium on*, pages 110–124, May 2005.

[9] Raphael C-W Phan and Patrick Mingard. Analyzing the secure simple pairing in bluetooth v4. 0. *Wireless Personal Communications*, 64(4):719–737, 2012.

[10] Claudio Soriente, Gene Tsudik, and Ersin Uzun. Hapadep: human-assisted pure audio device pairing. In *Information Security*, pages 385–400. Springer, 2008.

[11] Serge Vaudenay. Secure communications over insecure channels based on short authenticated strings. In *Advances in cryptology–CRYPTO 2005*, pages 309–326. Springer, 2005.