ONE-TOUCH FINANCIAL TRANSACTION AUTHENTICATION

Daniel V. Bailey, John Brainard
RSA, the Security Division of EMC, Bedford, MA U.S.A.

Sebastian Rohde, Christof Paar
Horst-Görtz Institute for IT Security, Ruhr-Universität Bochum, Germany

Keywords: User authentication, Online banking, Wi-Fi.

Abstract: We present a design for a Wi-Fi user-authentication token that tunnels data through the SSID field, packet timing, and packet length. Previous attempts to build an online-banking transaction-signing token have been only moderately successful, due in large part to usability problems. Average consumers, especially in the United States, are simply unwilling to transcribe strings of digits from PC to token and back again. In a departure from previous work, our token communicates using point-to-point side-channels in Wi-Fi that allow two devices to directly exchange messages – even if one is also connected to an access point. The result is a token that can authenticate transactions using only one touch by the user. The increased usability means more transactions can be authenticated, reducing fraud and driving more banking business online.

1 INTRODUCTION

The contribution of this paper is a design for a Wi-Fi user-authentication token with a focus on online banking. In its most basic form, our token tunnels authentication data through the SSID field found in Wi-Fi management frames. Our new techniques allow the seamless operation of three devices: a specialized token, a standard Access Point (AP) and a PC with application-layer software to interact with the token. To that end, this paper introduces a constructive use of Wi-Fi side channels. Although side channels are usually treated in the literature as a means to attack systems, in this paper we propose their use as a legitimate communication channel.

For its part, Wi-Fi represents both an opportunity and a challenge. It is widespread, especially in PCs. Because it suffers from several disadvantages, we propose new methods of co-opting Wi-Fi using certain data fields, packet length, and packet timing, to overcome these limitations.

Our work is motivated by the difficulties of user-authentication schemes for retail online banking. We equip users with a wireless device which we'll call the token. The token is used during login and transaction execution. Although authentication tokens have been available for years, they have historically imposed a serious burden on users. Our new design addresses this shortcoming by using wireless communication.

Because we aim to use this device on a broad range of consumer PCs as well as embedded systems, Wi-Fi support is a top priority. At least in the United States, Bluetooth isn't supported by most laptops used by consumers. Although in principle one could outfit users with a USB-to-Bluetooth adapter, requiring the user to manage two pieces of hardware (adapter and token) would defeat our usability goals.

2 RELATED WORK

The use of PCs with handhelds has been seen as a way to secure user input and output. The Pebbles project uses a handheld as an additional user interface for a PC (Myers, 2001), as does the Apple Remote Control (Apple, 2008). In "Bump in the Ether" (McCune et al., 2006), the authors leverage trusted input and output on the handheld to compensate for the possibility of malware on the PC. The authors of "Handheld Computers can be Better Smartcards" (Balfanz and Felten, 1999) similarly use the assumption of trusted output on the handheld screen to allow the user to verify data before signing.

V. Bailey D., Brainard J., Rohde S. and Paar C. (2009).
ONE-TOUCH FINANCIAL TRANSACTION AUTHENTICATION.
In *Proceedings of the International Conference on Security and Cryptography*, pages 5-12
DOI: 10.5220/0002182400050012

The "Zero-Interaction Authentication (ZIA)" paper by Corner and Noble (Corner and Noble, 2002), envisions a Wi-Fi token worn by users. That work focuses on encrypting data held on the PC with the keys supplied by the token as needed. This scheme requires essentially continuous operation of the token's radio and processor leading to high power consumption. The difference with our work illustrates our contribution. The ZIA token uses ordinary Wi-Fi association and UDP for data transmission and reception. This exceptionally high duty cycle leads in practice to a token with power consumption that is unacceptably high. In addition, this design makes roaming difficult: in order for the PC to communicate both with Internet services and the token, both would need to associate and communicate through the same access point.

In a similar vein, in the "Zero-Stop Authentication" paper by Matsumiya, et al (Matsumiya et al., 2005), the system designers aim to imbue the physical environment with Wi-Fi radios and RFID readers to automatically engage in a challenge-response protocol with the user's cellphone or PDA. "Context-Aware User Authentication" by Bardram, et al (Bardram et al., 2003) also aims to combine location data into an overall authentication decision.

A commercial offering from Ozmo, in collaboration with the Intel Cliffside project, has been announced (Merritt, 2008). At the time of this writing, little is known about how the system actually works. From the available high-level description, their goals are similar to ours: an alternative use of Wi-Fi for point-to-point networking. Their solution requires rewriting of network drivers; our solution needs only application-layer software with no chipset-by-chipset customization. Ultimately, Ozmo should be seen as a complementary solution: we could use it for data transmission and reception.

3 IMPEDIMENTS TO WI-FI USE IN ONLINE BANKING TOKENS

The model of a standard PC using Wi-Fi communicating to an AP equipped with a backhaul connection to the Internet is now ubiquitous among online-banking customers worldwide, and we aim to devise a security token consonant with this use case. This mode of operation, called "infrastructure mode," has the AP intermediate all traffic. As defined in Section 7.2.2 of (IEEE, 2007), even if two PCs are in close proximity, data frames are sent from the PCs to the AP to be relayed, and never directly to one another.

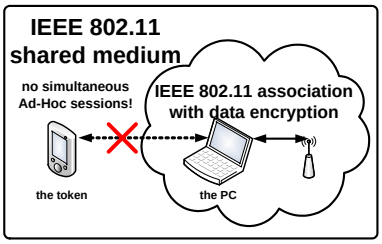

Figure 1: IEEE 802.11 communication limits.

3.1 Single-session Only

Wi-Fi offers "ad-hoc mode" in which two PCs may directly communicate, but after testing a broad range of wireless cards (ten different cards representing five different chipset vendors) on Windows and Linux, we found that none were capable of holding simultaneous ad-hoc and infrastructure mode sessions, as depicted in Figure 1. This fact means that for practical purposes, a PC must communicate directly with the token, or with the AP (and, by extension, the online bank), but not both simultaneously.

3.2 Lengthy Wi-Fi Association Process

Unlike a PC, an authentication token is used infrequently, say to initially log in to the bank or to sign a transaction. If a token must register to an AP, a designer faces two unappealing choices: either the token registers with the AP once and continuously runs its radio to remain registered during the PC's entire session, or the token automatically re-registers on-demand. Re-registration introduces unacceptably long delays while the user waits for a response. The other choice - continuously running the radio - quickly drains the battery.

3.3 Key Distribution for MAC-layer Security

The token could be provisioned with access credentials to associate with the AP and complete 802.1X authentication (hereafter, we'll abuse the terminology and refer to these two separate processes simply as "registration") just like the PC. Unfortunately, the registration process varies among APs, which becomes particularly problematic among mobile banking users who may use fee-for-service hotspots. Some use a static WEP/WPA passphrase which would need to be manually entered into the token, defeating our usability goals. Others use 802.1X, for which a username and password would be required. Still others rely on an IPSec VPN.

3.4 Our Contribution

To enable bidirectional communications in the token, we propose the use of side channels to *simulate* simultaneous sessions, independent of Wi-Fi association or MAC-layer encryption, as depicted in Figure 2. We will propose a number of unidirectional side channels which may be used together to implement cryptographic protocols with a sharp application focus on online banking. The side channels will be constructed by finding ways to tunnel security-protocol data through standard 802.11 frames.

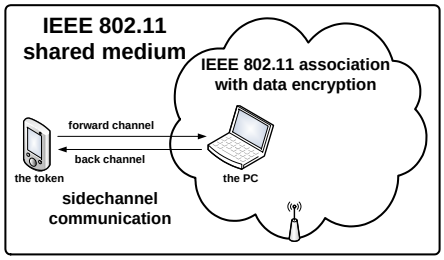

Figure 2: Our design.

4 BRIEF SURVEY OF ONLINE BANKING AUTHENTICATION

Several major approaches to user authentication for online banking are currently in use. User acceptance and a bank's tolerance for fraud typically guide these procurement choices. In this section, we survey the major authentication mechanisms in use with banking websites. We focus on the susceptibility to protocol-level attacks as well as usability characteristics.

4.1 The Threat

Strong user authentication is a critical component of an online banking application. The failure of the user authentication component can allow an attacker to perform an *account takeover* attack. In this form of online fraud, the attacker steals login credentials, such as passwords and proceeds to log into a legitimate user's banking account. Afterward, the attacker can steal by transferring money to his own account. This attack is so profitable in the real world that criminals have evolved sophisticated organizations to steal and evade detection and prosecution.

4.2 Static Passwords

Despite widespread shortcomings, the static password remains quite common for online banking, especially in the United States. The main attractions are clear: very low-cost provisioning and no need for the user to carry another device. The downsides, however, are well-documented. Passwords can often be guessed by the adversary or obtained through social engineering.

Phishing attacks result in a perfect clone of a static password. In their most familiar form, these amount to a bogus-website attack. The attacker assembles a website that looks similar to that of a legitimate bank. Then the attacker sends mass emails enticing users to visit the fake website and provide their password which can be later replayed. Each of the major alternative approaches to online banking authentication aims to disrupt this protocol attack and strike a subtle balance between usability and security.

4.3 TAN Lists

An alternative is for the bank to supply the user with a printed list of passwords called a Transaction Authentication Number (TAN) List. This numbered list of one-time passwords allows a form of challenge-response protocol: the bank website asks the user for a password located at a random position on the list with each challenge used only once.

At the protocol level, introducing a security artifact - even one as simple as a list of passwords - introduces a potential attack vector. If the user leaves the list unattended, an attacker can clone it using a photocopier and return the original intact to the user.

Relying on physical possession to deter cloning is an improvement in practice over the cloning resistance of a static password which often can be simply guessed. Moreover, it thwarts the most common phishing attacks since the adversary does not know which password the server will request on the next login attempt. More-sophisticated forms of phishing called active, spear-phishing, or man-in-the-middle (MITM) attacks still succeed (Parno et al., 2006). In these scenarios, the attacker intercepts the password and replays it to the bank in real time.

The list of passwords also harms usability. Without the password list, the user cannot log in. For online banking, any decrease in usability means less business transacted online, translating into less profit for the bank. If measures to increase security harm usability, few banks are interested.

4.4 Handheld OTP Tokens

Traditional handheld one-time password (OTP) tokens are today available from commercial vendors. In general, these are handheld devices with a small LCD that shows changing passwords. Standardization has

begun in the IETF, including (M'Raihi et al., 2005; M'Raihi et al., 2008a; M'Raihi et al., 2008b). Unlike a printed password list, the device throttles the display of passwords either using an internal clock or a button, making it harder to clone. MITM attacks can still succeed, as the adversary can intercept a password and instantly replay it to the website.

An OTP token is still an artifact that must be carried by the user, who in turn must transcribe digits into the PC – both harming usability.

4.5 Keypad Signing Tokens

The MITM attacks on OTP tokens are a product of the lack of direct linkage between the generation and consumption of passwords. Authenticating the user only at login time means that a MITM attacker can intercept and instantly replay passwords. Once logged in, the attacker can fraudulently transfer funds. To tie passwords directly to financial transactions (instead of user sessions), some tokens are equipped with keypads to accept external inputs. The user enters transaction details like transaction ID, account numbers, transaction amount and/or random challenge directly into the token. In response, the signing token uses a symmetric key shared with the server to compute and display a truncated message authentication code. The user then enters the message authentication code into the PC which relays it on to the banking server. Observe that relying on the user to transcribe digits in this way offers a protocol advantage: she has an opportunity to validate the transaction details. If an attacker tries to introduce say a fraudulent account number or transaction amount, the user can recognize this fact and decline to enter it into the token. Merely intercepting the resulting authentication code does not allow the attacker to fraudulently transfer funds. Modifying the account number would invalidate the authentication code. Unfortunately, this improved protocol performance comes at the price of reduced usability. If users already complain about transcribing six-to-eight digit passwords, then transcribing digits from the PC to the token and back for each transaction is even worse.

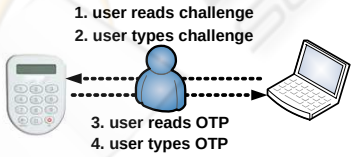

Figure 3: Keypad signing token.

4.6 Smart Cards

Smart cards offer a bidirectional machine interface in a set of electrical contacts. Because smart-card readers are not standard equipment on most PCs, the user must keep track of two pieces of hardware: card and reader. This requirement makes it difficult to log in away from home or office, such as at a kiosk. But like the signing tokens, one can authenticate transactions to thwart network-based MITM attacks.

The direct PC data-interface, however, leads to a lack of direct agency on the part of the user and a new MITM vulnerability, described as the "rogue terminal" problem in (Schneier and Shostack, 1999). The potential avenue for attack lies in the lack of trusted I/O between the smart card and the user. In contrast to our previous MITM where the attacker is active on the network, here the attacker is active on the PC-to-smart card channel. For instance, malware on the PC could display the user's requested transaction details while providing fraudulent details to the smart card for signature. A standard countermeasure is to collect a PIN from the user, but unfortunately if malware is present on the PC, it can intercept and replay the PIN.

To thwart this attack one needs a direct interface on which the user can confirm transaction details before a signature is computed. Some banks provide a smart-card reader with keypad and display for this purpose, but this is expensive and requires the user to manage two pieces of hardware. Nevertheless, smart cards are often engineered to resist certain forms of physical and side-channel attack, which is more difficult with our software-only solution.

5 THE TOKEN-PC "FORWARD CHANNEL"

Our solution aims for the same protocol security as the smart card solution, but using only PC and mobile phone software. In addition, the user should have these protocol guarantees without typing digits.

By exploiting different frames, we will construct two unidirectional channels: one from the token to the PC which we call the "forward channel," and a channel from the PC to the token which we call the "back channel." We assume the PC runs Windows XP or Vista and is already using its Wi-Fi adapter in an infrastructure session. There are three possible token configurations: those capable of *neither* monitor mode nor packet injection; monitor mode *but not* packet injection; and *both* monitor mode and packet injection. We will treat two possible cases for the token: those capable of both monitor mode and packet

injection and those capable of neither, as we believe these are the two most common cases in practice.

The token-to-PC forward channel does not require any special modes of Wi-Fi and may therefore be implemented on any device. To construct the forward channel, we exploit the methods by which 802.11 clients determine which wireless networks in range offer service. Both APs and ad-hoc network participants advertise their existence by sending out *beacon frames*, which are unsolicited management frames sent to the broadcast address; and *probe response frames*, which are sent in response to *probe frames* sent out by stations seeking access points.

These two data paths define two different methods for a PC to obtain information about nearby APs: passive scanning, wherein a station merely listens for beacons, and active scanning where a station sends probe requests and listens for probe responses. Passive scanning is used in the popular Kismet wireless network-detection software (Kershaw, 2004). Like Kismet, our experiments confirmed that few network adapters are capable of passive scanning on Windows, with most only supporting active scanning, and none supporting only passive scanning. Building a widely-usable forward channel thus requires using active scanning on the PC. We wrote a simple user-mode PC application that scans for APs, sending out probe requests and listening for probe responses.

A probe response frame contains the 32-byte SSID field, which typically indicates the natural language name of a network. The SSID field can be given an arbitrary value and still be propagated unmolested up the protocol stack by commodity 802.11 hardware and Windows drivers.

After testing different Wi-Fi PC adapters (from Linksys, D-Link, Belkin, Hawking, Netgear, and Intel) with different chipsets (from Prism, Ralink, Atheros, Broadcom, and Intel), we found that only the SSID field can be reliably received by the PC in this way. Other fields we tested were unsuccessful. Vendor specific fields of up to 255 bytes can also be appended to probe response frames and used to carry authentication data. Unfortunately, one adapter only reported the first 240 bytes and others did not report these other fields at all.

Because of these limitations, our authentication token thus sends messages by essentially advertising itself as an ad-hoc network participant, sending out beacon frames and probe response frames upon activation with the SSID field set to the base-64 ASCII encoding of the cryptographic message. The SSID field identifies the frame as part of the authentication protocol using the ";" character (an arbitrary choice) and carries the cryptographic payload necessary to authenticate the token to the PC. Software on the PC continuously searches for access points and ad-hoc network participants; when one is found whose SSID begins with the ";" character, it delivers the cryptographic payload to security applications. The delay from when the prototype begins to advertise the ad-hoc network to the time the payload is passed to application software on the mobile varies depending on what 802.11 chipset and driver is used on the mobile. We experimented the most with a Belkin F5D7050 ver. 3; using it, the delay is around three seconds.

The SSID field is at most 32 bytes long; the ";" character consumes one byte leaving 31 bytes. This is a very small payload, but much larger than user-transcribed OTPs which amount to around 33 bits. For fragmented messages, the token can masquerade as multiple access points simultaneously.

6 THE PC-TOKEN "BACK CHANNEL"

Once associated with an AP using MAC-layer encryption, the operating system does not offer an API to enable or disable encryption on a frame-by-frame basis. For this reason, application software running on the PC cannot directly send cryptographic challenges or transaction details in data frames that can be read by the token. Because the token does not have the MAC-layer encryption keys, even if its drivers support monitor mode, the best the token can hope for is to intercept encrypted frames, shown in Figure 2.

Regardless of the type of MAC-layer encryption in use, the token can expect to observe some properties of the encrypted data frames. In particular, the size of data frames can be seen by an adapter in monitor mode. In addition, Wi-Fi encryption is applied only to the data payload of a frame and not to its headers. Fortunately, Wi-Fi interfaces provide the ability to set certain unencrypted header fields from application-layer software. In this section, we will explore the use of these side-channels to construct a back channel from the PC to the token to deliver challenge or transaction details. The token must be capable of monitor mode operation in order to use one of these back channels. Personal wireless devices including the Nokia Internet Tablet offer this feature and naturally any purpose-built token would as well.

6.1 The MAC-Channel

Because the PC usually is already engaged in an infrastructure session – and most chipsets are single-session only – its Wi-Fi stack is not able to di-

rectly address the token. Using only application-layer software to create explicitly malformed Ethernet-II frames, a PC using Windows or Linux can be tricked to set certain header fields of the IEEE 802.11 protocol to arbitrary values, allowing us to create a side-channel that can be received by any phone or wireless device with monitor-mode capability. These frames can therefore be used as a side channel to transfer data out of a Wi-Fi network.

6.2 The Length-Channel

The other option for the PC to transport data to the token is to use the lengths of packets as a side channel. This alternative approach is valuable in cases where the operating system does not easily support the MAC-Channel, or could be combined with the MAC-Channel to increase throughput. A simple encoding is used to translate the data into a sequence of packet lengths. n distinct packet lengths provide an entropy of $e = log_2(n)$ bits of information. Sending one packet of length l_i with $i = 1..n$ will transport e bits of information.

7 IMPLEMENTATION

The forward channel is compatible with every driver and chipset combination we tested. By contrast, the backchannels require the token to operate in monitor mode – only available for certain driver and chipset combinations. Nevertheless, the proliferation of Linux in personal wireless devices makes monitor mode available in mainstream devices such as the Nokia Internet Tablet N810. Our token implementation was tested on both a Linux PC and the N810. The implementation relies on a specialized library called Lorcon (LORCON, 2008) to support monitor mode. It also provides helper methods to support injection of raw 802.11 packets.

7.1 The SSID Channel from Token to PC

From the point of view of software running on the token, there are two possible ways to establish a side channel using the SSID field. The first is to let the network interface act as a regular access point. We experimented with the Belkin F5D7050 Version 3 adapter, which uses the Ralink RT73 chipset, and the Atheros-based Netgear WG511T adapter. We found that the driver (MadWifi, 2008) for the Atheros-based card is natively capable of emulating two access points simultaneously, but more lead to a kernel crash. To emulate more access points, the implementation resorts to packet injection. Section 8.1 elaborates on the performance impact of simultaneous SSIDs. The SSID field can transport 32 bytes. Because some drivers have difficulty with non-ASCII characters, the token uses base64 encoding. As previously mentioned, the first character is ";". The second is a monotonically increasing packet counter, necessary for the reconstruction of fragmented messages and detection of lost fragments. The token software responds directly to 802.11 probe requests and sends out beacons. Each virtual access point is announced for five seconds so the software on the PC has a chance to receive data.

The receiving part of the SSID channel on the PC uses the wireless API introduced in Windows Vista and Windows XP SP3 to listen continuously for new networks. If new networks match the discriminator, they are accepted and if necessary, defragmented.

7.2 The Length Channel from PC to Token

As described in Section 6, the length of packets transferred over the wireless interface is used to encode information. Therefore the PC needs to be able to send packets and control their length which can easily be done by using the library libnet(Libnet, 2008) which in turn relies on WinPCap(WinPcap, 2008). The receiving part for the token only needs to look at the length of certain encrypted packets. The recipient has no direct way to distinguish side-channel packets from those that are created through regular network traffic. However, by reserving certain packet lengths to serve as start-of-block and end-of-block delimiters, one can decrease the likelihood of misinterpreted packets. This sequence also has some basic encoding to detect if some packets have been falsely interpreted. The data is divided into fixed-length blocks. Every n bits of every block is encoded into one out of $e = 2^n$ packet lengths. To provide flow control, the encoded blocks are surrounded by packets of length corresponding to the reserved start- and end-of-block values. By counting the number of packets in a block, the PC can detect either extraneous or lost packets.

One could apply sophisticated codes to detect and repair errors; our purpose here is merely to implement and report on the basic channel characteristics to which coding could be applied.

7.3 The MAC Channel

The basic mechanism of the MAC Channel is comparatively simple. The sending part forges raw Ethernet

II frames through the libnet packet factory. The destination field of the protocol header is used to encode data, providing six bytes of data per packet. In addition the packets are sent with only one byte of payload. The Windows networking stack in conjunction with the appropriate drivers will just translate these fields into the corresponding IEEE 802.11 data fields.

This results in a very unusual packet length which is used to distinguish these side-channel packets from other network traffic. To avoid problems caused if the destination matches an existing MAC address the Ethernet II protocol field is set to an invalid value.

7.4 Experimental Setup

The Nokia Internet Tablet N810 is capable of providing the SSID channel and monitor mode without modifications, but not arbitrary wireless packet injection – which yields higher performance. For this feature we turn to an external USB Wi-Fi adapter. The N810 development kit includes the complete kernel source which in turn allows one to easily compile the RT73 open source driver (RT73, 2008) for the Belkin FD5D7050 Version 3 Wi-Fi USB adapter. Because the USB port of the tablet only provides a limited amount of current, we also use a powered USB hub between the N810 and the Wi-Fi adapter. We use this hardware setup to measure the channel performance by sending a fixed amount of data through the various side channels.

8 PERFORMANCE MEASUREMENTS

This section reports on channel delay, throughput and the loss rate of the proposed side channels. The channel delay is measured by comparing the time when data is sent and the time when data arrives. Because of the amount of time needed by the PC to scan for APs, it is around one to four seconds for the SSID channel. The delay for the other channels is close to the delay of the wireless channel itself as only a few ordinary packets are needed to deliver data to the channel's user. The loss rate is the percentage of bytes lost in transmission. Throughput denotes the number of uncompressed bytes per second and the loss rate gives information about the reliability of the channel. Table 1 shows the results. In the "No other traffic" scenario, we made no effort to generate additional load on the AP. For the "Multiple YouTube Video Downloads" scenario, we launched four simultaneous downloads to generate some traffic with which the side channels would have to compete. Finally, for the "SMB File Transfer," we initiated a large file download. In each scenario, we averaged over four experiments.

Table 1: Channel Performance with $n_{SSID} = 4$ and $e = 8$.

Channel	Throughput (bytes/s)	Loss Rate
No other traffic		
SSID-Channel	19	0.0%
Length-Channel	80	7%
MAC-Channel	2159	1%
Multiple YouTube Video Downloads		
SSID-Channel	10	40%
Length-Channel	32	19%
MAC-Channel	1754	7%
SMB File transfer		
SSID-Channel	8	50%
Length-Channel	30	20%
MAC-Channel	281	10%

Table 2: SSID Channel Parameter Evaluation.

Channel	Throughput (bytes/s)	Loss Rate
No other traffic		
$n_{SSID} = 4$	19	0.0%
$n_{SSID} = 8$	40	0.0%
$n_{SSID} = 16$	85	0.0%
$n_{SSID} = 32$	195	0.0%

8.1 Channel Parameters

The SSID and Length channel provide configuration options to change the behavior of the channel. The SSID channel can be configured by setting the maximum number n_{SSID} of simultaneously used SSIDs; each is transmitted for five seconds. Unfortunately the increased speed offered by simultaneous SSIDs comes at a price: large numbers of virtual access points may confuse both the end-user and the networking stack of their PCs. Table 2 shows how n_{SSID} affects the channel characteristics.

The length channel can be configured by the number of distinct packet lengths n that are used. Besides some overhead the performance is theoretically proportional to the bits of information $e = log_2(n)$ that are encoded in one packet length. For simplicity we measured $e = 2$, 4, 8 yielding (together with one delimiter) 5, 17, 257 distinct packet lengths. The throughput can be increased by choosing shorter packet lengths. We chose packet lengths starting from 500 bytes because due to Lauradoux (Lauradoux, 2007) they are very uncommon in normal TCP traffic and therefore less likely to be mistaken for ordinary network traffic. In addition, it eases the burden on the decoder: the packet lengths represent two bits, four

Table 3: Length Channel Parameter Evaluation

Channel	Throughput (bytes/s)	Loss Rate
No other traffic		
$e = 2$	22	17%
$e = 4$	46	13%
$e = 8$	80	7%

bits, and eight bits respectively. The channel characteristics for the parameterized length channel can be found in Table 3. The loss rate decreases with an increasing number of lengths because the number of packets sent per data frame is decreased.

9 CONCLUSIONS

By necessity, authentication tokens have traditionally implemented one-way cryptographic protocols. Expanding the data channel beyond digits typed by a human expands both the possibilities and risks. Fast bidirectional communication further ups the ante, allowing fine-grained data-origin authentication and document signing with a robust human interface including sensors, biometrics, and other complementary authentication factors. These robust functions must continue to be balanced against the severe power consumption constraints in mobile devices. We leave the protection of the token-PC link for future work. Careful modeling of authentication and key-distribution (pairing) protocols is needed.

REFERENCES

Apple (2008). About the apple remote control. Available at http://support.apple.com/kb/HT1522.

Balfanz, D. and Felten, E. (1999). Hand-Held Computers Can Be Better Smart Cards. *8th USENIX Security Symposium*, 271.

Bardram, J., Kjær, R., and Pedersen, M. (2003). Context-Aware User Authentication–Supporting Proximity-Based Login in Pervasive Computing. *Proceedings of Ubicomp*, pages 107–123.

Corner, M. and Noble, B. (2002). Zero-interaction authentication. *Proceedings of the 8th annual international conference on Mobile computing and networking*, pages 1–11.

IEEE (2007). IEEE 802.11-2007. IEEE standard for information technology–telecommunications and information exchange between system–local and metropolitan area networks specific requirements–part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications.

Kershaw, M. (2004). Kismet. Referenced 2008 at http://www.kismetwireless.net/presentations/5hope-kismet.pdf.

Lauradoux, C. (2007). Throughput/code size tradeoff for stream ciphers. The State of the Art of Stream Ciphers - SASC.

Libnet (2008). The libnet packet construction library. available at http://www.packetfactory.net/libnet/.

LORCON (2008). Lorcon (loss of radio connectivity). available at http://802.11ninja.net/lorcon.

MadWifi (2008). Madwifi wlan driver. available at http://madwifi.org/.

Matsumiya, K., Aoki, S., Murase, M., and Tokuda, H. (2005). A zero-stop authentication system for sensor-based embedded real-time applications. *J. Embedded Comput.*, 1(1):119–132.

McCune, J. M., Perrig, A., and Reiter, M. K. (2006). Bump in the ether: A framework for securing sensitive user input. In *Proceedings of the 2006 USENIX Annual Technical Conference*, page 185198.

Merritt, R. (2008). Wi-fi jumps into the pan. *EETimes*, June 6th, 2008. Available at http://www.eetimes.com/news/latest/showArticle.jhtml?articleID=208401238.

M'Raihi, D., Bellare, M., Hoornaert, F., Naccache, D., and Ranen, O. (2005). Hotp: An hmac-based one-time password algorithm. http://www.ietf.org/rfc/rfc4226.txt.

M'Raihi, D., Machani, S., Pei, M., and Rydell, J. (2008a). Totp: Time-based one-time password algorithm. http://www.ietf.org/internet-drafts/draft-mraihi-totp-timebased-00.txt.

M'Raihi, D., Rydell, J., Naccache, D., Machani, S., and Bajaj, S. (2008b). Ocra: Oath challenge-response algorithms. http://www.ietf.org/internet-drafts/draft-mraihi-mutual-oath-hotp-variants-07.txt.

Myers, B. (2001). Using handhelds and PCs together. *Communications of the ACM*, 44(11):34–41.

Parno, B., Kuo, C., and Perrig, A. (2006). Phoolproof Phishing Prevention. *LECTURE NOTES IN COMPUTER SCIENCE: Tenth Financial Cryptography and Data Security Conference*, 4107.

RT73 (2008). The rt73 driver homepage. available at http://rt2x00.serialmonkey.com/.

Schneier, B. and Shostack, A. (1999). Breaking Up is Hard to Do: Modeling Security Threats for Smart Cards. *USENIX Workshop on Smartcard Technology*.

WinPcap (2008). Winpcap: The windows packet capture library. available at http://www.winpcap.org/.