Secure Device Pairing based on a Visual Channel *

Nitesh Saxena†
University of California, Irvine, USA
nitesh@ics.uci.edu
Jan-Erik Ekberg, Kari Kostiainen, N. Asokan
Nokia Research Center, Helsinki, Finland
{jan-erik.ekberg, kari.ti.kostiainen, n.asokan}@nokia.com

Abstract

Recently several researchers and practitioners have begun to address the problem of *secure device pairing* or how to set up secure communication between two devices without the assistance of a trusted third party. McCune, et al. [12] proposed Seeing-is-Believing (SiB), a system which uses a visual channel. The SiB visual channel consists of one device displaying the hash of its public key in the form of a two-dimensional barcode, and the other device reading this information using a photo camera. Strong mutual authentication in SiB requires running two separate unilateral authentication steps.

In this paper, we show how strong mutual authentication can be achieved even with a uni-directional visual channel, where SiB could provide only a weaker property termed as *presence*. This could help reduce the SiB execution time and improve usability. By adopting recently proposed improved pairing protocols, we propose how visual channel authentication can be used even on devices that have very limited displaying capabilities, all the way down to a device whose display consists of a cheap single light-source, such as an LED. We also describe a new video codec that may be used to improve execution time of pairing in limited display devices, and can be used for other applications besides pairing.

1 Introduction

The popularity of short-range wireless technologies like Bluetooth and Wireless Local Area Networking (WLAN) based on the IEEE 802.11 family of protocols is experiencing enormous growth. Newer technologies like Wireless Universal Serial Bus[1] are around the corner and promise to be as popular. This rise in popularity implies that an ever increasing proportion of the users of devices supporting short-range wireless communication are not technically savvy. Such users need very simple and intuitive methods for setting up their devices. Since wireless communication is easier to eavesdrop on and easier to manipulate, a common set up task is to initialize secure communication. In this paper, we will use the term *pairing* to refer to this operation.[2]

*A shorter version of this paper appears in [16]

†Work done while visiting Nokia Research Center, Helsinki

[1] http://www.usb.org/developer/wusb

[2] The term *pairing* was introduced in the context of Bluetooth devices. Other roughly synonymous terms include "bonding," and "imprinting".

Consequently, both security researchers and practitioners have been looking for intuitive techniques for ordinary users to be able to securely pair their devices. Although the primary impetus comes from the need to secure short-range wireless communication, the issue of intuitive security initialization is more generally applicable whenever ordinary users need to set up secure communication without the help of expert administrators or trusted third parties.

The pairing problem is to enable two devices, which share no prior context with each other, to agree upon a security association that they can use to protect their subsequent communication. Secure pairing must be resistant to a man-in-the-middle adversary who tries to impersonate one or both of these devices in the process. The adversary is assumed to be capable of listening to or modifying messages on the communication channel between the devices. One approach to secure pairing is to use an additional physically authenticatable channel, called an out-of-band (OOB) channel which is governed by humans, i.e., by the users operating these devices. The adversary is assumed to be incapable of modifying messages on the OOB channel, although it can listen to them.

There has been a significant amount of prior work on building secure pairing protocols using OOB channels [17, 2, 5, 8]. They consider different types of OOB channels including physical connections, infrared, etc. Recently, McCune, et al. proposed a scheme called "Seeing-is-Believing" (SiB), where the OOB channel is implemented as a visual channel. The SiB visual channel consists of a two-dimensional barcode of [15], displayed by (or affixed to) a device A, that represents security-relevant information unique to A. A user can point another camera-equipped device B at the barcode so that B can read the barcode visually, and use this information to set up an authenticated channel to A. If both devices are camera-equipped, they can mutually authenticate each other. "Authentication" in this case is based on demonstrative identification [2] rather than with respect to a claimed name.

Our Contributions: In this paper, we propose several improvements and extensions to the SiB system. Our contributions are as follows:

1. We show how strong mutual authentication can be achieved using just a unidirectional visual channel. This results in two improvements:
   (a) strong authentication becomes possible in situations where SiB could only achieve a weaker property termed as "presence".
   (b) execution time for mutual authentication decreases significantly and usability improves.
2. By adopting recently proposed improved protocols [10], we show how visual channel authentication can be used even on devices that have very limited displaying capabilities, all the way down to a device whose display consists of a cheap single flashing light-source, such as a single light-emitting diode (LED).
3. We also propose a *video-based* codec which may help improve the speed of secure pairing in devices with constrained displays, as well as may lead to applications other than secure device pairing.

The rest of the paper is organized as follows. First, we start with a brief description of SiB in Section 2. In Section 3 we describe an alternative protocol that improves the presence guarantee provided by SiB to full-fledged mutual authentication. Then, in Section 4, we show how visual channel authentication can be done even in highly constrained environments. We discuss the applicability and relevance of our improvements and extensions in Section 5.

2 Seeing-is-Believing

Several researchers have proposed the idea of encoding service or device discovery information in the form of barcodes so that they can be read using camera phones [15, 3, 19, 11]. The idea

of encoding cryptographic secret material into barcodes was first proposed by Hanna [8] as well as Gehrmann, et al. [5], both of which also mention the use of asymmetric key cryptography in this context. The SiB paper [12] by McCune et al. was the first research paper to propose that the information encoded in the barcode could be a commitment to a public key.

In SiB, a device A can authenticate to a device B, if B is equipped with a camera. A's commitment to its public key (such as a hash) is encoded in the form of a two-dimensional barcode of [15]. A typical barcode has dimensions approximately 2.5x2.5 cm^2 to allow recognition from a reasonable distance, and consists of a total of 83-bits of information (68-bits of data and 15-bits for forward error correction). If A has a display, the public key can be ephemeral, and the barcode is shown on the display. Otherwise, A's public key needs to be permanent and the barcode is put on a printed label affixed to the housing of A. Authentication is done by the user pointing B's camera at A's barcode. The basic unidirectional authentication process is depicted in Figure 1.

1. A calculates h_A as $h(K_A)$

   $A \longrightarrow B$ (visual channel): h_A

2. $A \longrightarrow B$ (insecure channel): K_A

   B calculates h' as $h(K_A)$ using the K_A received. If h' does not match the h_A received in Step 1, B aborts.

Figure 1: SiB unidirectional authentication protocol (B authenticates A)

K_A is A's public key. $h()$ is a cryptographic hash function, which is resistant to second pre-image finding. K_A can be long-lived, in which case the output of $h()$ must be sufficiently large, e.g., at least 80-bits. If K_A is ephemeral, the output of $h()$ can be smaller, e.g. 48 bits [6]. SiB could accommodate 68 bits of hash into a single two-dimensional barcode, but requires a good quality display due to the typical size of the barcode[3]. Mutual authentication requires the protocol of Figure 1 being run in each direction. This has two implications for SiB.

- First, mutual authentication is possible only if **both** devices are equipped with cameras. McCune, et al. state (Section 7 of [12])

  > A display-only device ... is unable to strongly authenticate other devices using SiB ... [because it] cannot "see" them.

  If a device A is not equipped with a camera, it can only achieve a weaker property known as "presence," by including a secret key K in the barcode. The camera-equipped device B that reads the barcode can use K to compute message authentication code (MAC) over the message it sends to A. If the MAC is correct, A can conclude that it was sent by some device that was able to "see" its barcode, and thus was "present". Presence is a weaker security notion than authentication because A has no means of knowing if B is really the device that the user of A intended to communicate with.

  We summarize the types of authentication achievable using SiB for given combinations of device types in Table 1.

- Second, in order to run the protocol in each direction, the roles of the devices have to be switched so that first A's camera can scan B's display and then B's camera can scan A's display. Such switching of devices by users not only increases the execution time of the SiB process but also decreases usability. McCune, et al. report that the average SiB

[3] SiB can encode the data into several barcodes displayed in sequence.

execution time in their user trials was 8 seconds, even though time required to recognize a barcode is just about one second [15].

Y has $\rightarrow$ X has $\downarrow$	Camera and display	Camera only	Display only	None
Camera and Display	$X \leftrightarrow Y$	$X \leftrightarrow Y_s$	$X \leftarrow Y$ $X \xrightarrow{p} Y$	$X \leftarrow Y_s$
Camera only	$X_s \leftrightarrow Y$	$X_s \leftrightarrow Y_s$	$X \leftarrow Y$ $X \xrightarrow{p} Y$	$X \leftarrow Y_s$
Display only	$X \rightarrow Y$ $X \xleftarrow{p} Y$	$X \rightarrow Y$ $X \xleftarrow{p} Y$	none	none
None	$X_s \rightarrow Y$	$X_s \rightarrow Y$	none	none

Notation:
P_s: "Device P needs a static barcode label affixed to it."
$P \rightarrow Q$: "Device P can strongly authenticate to device Q."
$P \xrightarrow{p} Q$: "Device P can demonstrate its presence to device Q."

Table 1: Types of authentication achievable using SiB for given device type combinations

These implications limit the applicability of SiB in various practical settings. Many devices cannot have either cameras or high quality displays for different reasons. Commoditized devices like WLAN access points are extremely cost-sensitive and the likelihood of adding new hardware for the purpose of authentication is very small. Devices like Bluetooth headsets are typically too small to have displays or even to affix static barcode stickers.

To summarize, we identify the following drawbacks with the basic SiB scheme:

1. Mutual authentication is not possible unless both devices are equipped with cameras.
2. The overall execution time for mutual authentication is high, which impacts usability.
3. Applicability of SiB is limited in situations where one device has limited capabilities (e.g., small size, no camera, limited or no display at all).

In the rest of this paper, we describe how we can address each of these drawbacks.

3 Seeing Better: Upgrading Presence to Authentication

In this section, we address the issue of mutual authentication. Recall that we identified two shortcomings of SiB in this respect. First, SiB can provide mutual authentication only if *both* devices are camera-equipped. Second, the processing time for mutual authentication is high.

We observe that both of these drawbacks stem from the fact that mutual authentication is done as two separate unidirectional authentication steps. Therefore, we propose to solve both problems by performing mutual authentication in a single step by having each of A and B compute a *common* checksum on public data, and compare their results via a unidirectional transfer using the visual channel. Let us call this protocol VIC, for "Visual authentication based on Integrity Checking." (See Figure 2.)

The security of the authentication of A to B in VIC depends on the attacker not being able to find two numbers $X1$ and $X2$ such that $h(K_A, X1) = h(X2, K_B)$. This implies that if the attacker can learn K_B ahead of time, $h()$ needs to be collision-resistant. If K_B is transient (or

1. $A \longrightarrow B$ (insecure channel): K_A
2. $A \longleftarrow B$ (insecure channel): K_B

   A calculates h_A as $h(K_A|K_B)$ and B calculates h_B as $h(K_A|K_B)$
3. $A \longrightarrow B$ (visual channel): h_A

   B compares h_A and h_B. If they match, B accepts and continues. Otherwise B rejects and aborts. In either case, B indicates accept/reject to the user.
4. A prompts user as to whether B accepted or rejected. A continues if the user answers affirmatively. Otherwise A rejects.

Figure 2: VIC mutual authentication protocol

a nonce picked by B is appended to K_B in message 2 and in the calculation of h_A and h_B), it is sufficient for $h()$ to be resistant against second pre-image finding, since the attacker can no longer use any pre-computed collisions. The security of the authentication of B to A depends, in addition, on the user correctly reporting the comparison result reported by B back to A. (Note that the entities taking part in the protocol are always assumed to be honest.)

Because VIC needs only a unidirectional visual channel, it is now possible to achieve mutual authentication in the cases where SiB could only achieve presence. In addition, the execution time for mutual authentication and the user effort will be less since no device role switching is required anymore. Thus, VIC addresses the first two drawbacks of SiB identified in Section 2.

In Table 2, we summarize the types of authentication achievable using VIC for given combinations of device types. Notice that since the checksum is different for each instance of VIC, at least one device must have a display and that the static barcode labels cannot be used with VIC.

Y has $\rightarrow$ X has $\downarrow$	Camera and display	Camera only	Display only	None
Camera and Display	X $\leftrightarrow$ Y	X $\leftrightarrow$ Y	X $\leftrightarrow$ Y	none
Camera only	X $\leftrightarrow$ Y	none	X $\leftrightarrow$ Y	none
Display only	X $\leftrightarrow$ Y	X $\leftrightarrow$ Y	none	none
None	none	none	none	none

Notation:
P $\leftrightarrow$ Q: "Devices P and Q can strongly authenticate each other."

Table 2: Types of authentication achievable using VIC for given device type combinations.

4 Seeing With Less: Visual Channel in Constrained Devices

Now we turn our attention to the third drawback of SiB. In this section, we show how to enable visual channel authentication on devices with very limited (or tiny) displays and in the minimal case, with extremely constrained displays consisting of only single light source (or LED). These extensions are made possible by using key agreement protocols that require short authenticated integrity checksums. We begin by describing such protocols.

4.1 Authentication Using Short Integrity Checksums

The reason why SiB needs good displays is the high visual channel bandwidth required for the SiB protocol. Assuming that the attackers have access to today's state-of-the-art computing resources, the bandwidth needed is at least 48 bits in the case of ephemeral keys [6], rising to 80 bits in the case of long-lived keys. These numbers can only increase over time.

Fortunately, there is a family of authentication protocols that has very low bandwidth requirements. The first protocols in this family, proposed by Gehrmann et al. in [5, 6], were aimed at using the human user as the authentication channel; hence the name "Manual authentication (MANA)". Several subsequent variations on the same theme have been reported [9, 18, 10]. We apply the variation called "MA-3" [10] to get VICsh (VIC with short checksum) as shown in Figure 3[4]:

1. A chooses a long random bit string R_A and calculates h_A as $h(R_A)$.
   $A \longrightarrow B$ (insecure channel): h_A, K_A
2. B chooses its own long random bit string R_B
   $A \longleftarrow B$ (insecure channel): R_B, K_B
3. $A \longrightarrow B$ (insecure channel): R_A
   B now computes h'_A as $h(R_A)$ and compares it with the h_A received in message 1. If they do not match, B aborts. Otherwise B continues.
4. A calculates hs_A as $hs(R_A, R_B, K_A, K_B)$ and B calculates hs_B as $hs(R_A, R_B, K_A, K_B)$
   $A \longrightarrow B$ (visual channel): hs_A
   B compares hs_A and hs_B. If they match, B accepts and continues. Otherwise B rejects and aborts. In either case, B indicates accept/reject to the user.
5. A prompts user as to whether B accepted or rejected. A continues if the user answers affirmatively. Otherwise A rejects.

Figure 3: VICsh mutual authentication protocol based on short integrity checksum

K_A, K_B are as in the case of SiB. $h()$ represents a commitment scheme and $hs()$ is a mixing function with a short n-bit output (e.g., $n = 15 \ldots 20$) such that a change in any input bit will, with high probability, result in a change in the output. In practice, $hs()$ can be the output of a cryptographic hash function truncated to n bits. Refer to [10] for formal description of the

[4]We chose MA-3 over the protocol in [18] for reasons of efficiency because MA-3 requires fewer rounds of communication over the insecure channel.

requirements on $h()$ and $hs()$, and their instantiations, as well as for the proofs of security of the protocol. Informally, the security of the protocol depends on the following:

- neither party reveals the value of its random bit string (R_A or R_B respectively) until the other party commits to its own random bit string, and
- each party knows that the public data (K_A and K_B) used in the computation of the check-value (hs_A or hs_B) is known to it before it reveals its random bit string.

Suppose the man-in-the-middle attacker has a public key K_M. To fool device A into accepting K_M as B's public key, the attacker needs to ensure that $hs_A = hs(R_A, X, K_A, K_M)$ and $hs_B = hs(Y, R_B, Z, K_B)$ are equal. The attacker can choose K_M, X, Y and Z, but he must make his choices before knowing R_A or R_B. Therefore, whatever his strategy for choosing the values, the chance of success is $x = 2^{-n}$. Similarly, the probability of the attacker fooling device B into accepting K_M as A's public key is also x. More importantly, this probability does not depend on the computational capabilities of the attacker, as long as $h()$ is secure.

4.2 Trimming Down the Display

Armed with the variation of VIC described above, we are now ready to investigate visual channel authentication on devices with very limited displays. Recall that our motivation is to support visual channel authentication on various commercial devices, such as wireless access points, Bluetooth headsets, etc. These devices typically have only the most limited form of a display consisting of a single bi-state light source, such as a single light-emitting diode (LED). In this section, we describe each aspect of the realization of single LED based visual channel authentication.

Transmission. We use frequency modulation to encode the data being transmitted (see Figure 4). The sender turns the light-source on and off repeatedly. The data is encoded in the time interval between each successive "on" or "off" event: a long gap represents a '1' and a short gap represents a '0'. Since the channel is unidirectional, the transmitter cannot know when the receiver starts reception. Therefore, the transmitter keeps repeating the sequence until either the user approves the key agreement, or a timeout occurs.

The camera phones of today are limited to a frame rate of about 10 video frames/second, and as we are receiving the bits with frequency modulation without synchronization, we are bound by the Nyquist-Shannon sampling theorem (sampling rate = 2 × bandwidth for no loss of information) [13]. This limits the transfer speed with this algorithm to around 5 bits/second.

Reception. The receiver processing is analogous: simplified, each received video frame is compressed into one value per frame (the sum of all the pixel values)[5], and the first-order difference between consecutive values (i.e., the derivative) is compared against a relative threshold based on maximum observed variation in the pixel sum. If the derivative is steep enough and in the right direction (alternating between positive and negative) a transition in lighting is registered. The time between two consecutive changes indicates the transfer of either a '1' or a '0' bit as depicted in Figure 4.

Trading Efficiency with Security. We designed two mechanisms that allow the possibility of a parameterizable trade-off between execution time and the level of security.

First, the data being transmitted via the visual channel, i.e., the integrity checksum, is known to the receiver in advance. We use this simple observation to reduce execution time. Recall that the sender repeats the n-bit string a number of times. The receiver proceeds in the

[5] The fact that the video frame is collapsed into one value per frame also shows the feasibility of using a sensitive light sensor combined with an analog-to-digital converter as a cheaper form of receiving device – with no change to the algorithms described in the paper. We have left the implementation of such a receiver as future work.

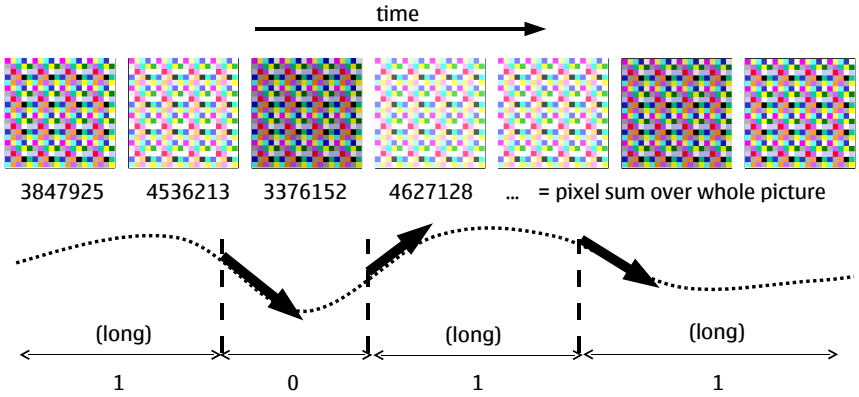


Figure 4: Data transmission via a single light-source visual channel

following way: reception may start at any bit position, and the receiver records until the n-bit tail of the received bit-string matches against any of the rotated versions of the expected n-bit string. Therefore, the receiver accepts at most n possible matches for the transmitted value. For example, if the transmitted string is '1011', the receiver accepts if it receives any of the strings '1011', '0111', '1110', '1101'.

Second, rather than doing error correction, we tolerate (or simply accept) a certain number of errors in the n-bit transmission. With k accepted errors, the number of possible matches, based on a binomial distribution of errors, is $\sum_{i=0\ldots k} \binom{n}{i}$.

Using these mechanisms the probability that the receiver will accept a random string as valid will increase from the original value of $p = \frac{1}{2^n}$. Accounting for both modifications we can estimate an upper bound to

$$p = n \, \frac{\sum_{i=0}^{k} \binom{n}{i}}{2^n}$$

The given bound allows us to get an idea of the degree of loss of security. If e.g. $k = 3$ bits are allowed to be wrong in an $n = 24$ bit sequence, p is 0.0064, whereas if only 1 bit error is allowed, p is 0.00004.[6]

For personal use, e.g., when a user wants to pair his workstation with his own wireless access point, an attack success probability of 0.00004 is acceptable. In other situations where, say, every day thousands of pairings are done with a device located in a public space, the attack success probability needs to be lower.

There are several ways to trade off security and execution time. The attack success probability p can be decreased by:

- increasing the length of the checksum n,

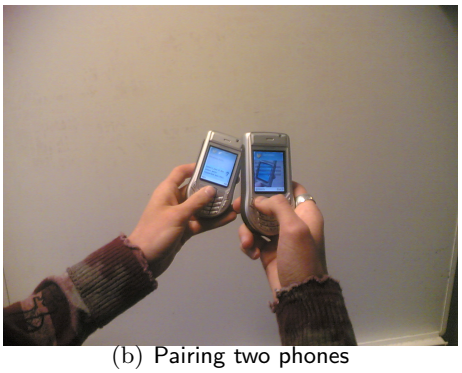
[6]In the pairing protocol case, the attack scenario is limited by the fact the the visual channel is authenticated, and the attacker is assumed to only operate in-band. In a more general case, where somebody might be feeding random visual data, the receiver also needs to check the signal history if the match is done later than after the first n received bits (with k errors). The history should give an indication that there is a repetition of the intended sequence (with possible errors) – if this is not the case the receiver is subject to a "visual attack" and accidentally found a match in a big sample of random input.

- reducing the number of acceptable errors k,
- reducing the number of possible rotations that are acceptable as matches (say only every fourth)
- adding an external end marker to the protocol (e.g., the light-source staying "on" for 0.5 seconds) to indicate when it starts to repeat the checksum string, bringing the attack success probability down to $\frac{\sum_{i=0}^{k}\binom{n}{i}}{2^n}$.

Applying one or several of these measures will result in changed lower and upper "bounds" for the execution time.

Implementation and Timings. We have developed a proof-of-concept implementation where a single blinking LED (connected to the parallel port of a PC) sends a signal that is received by a camera phone. Figures 5(a) and 5(b) illustrate our two demonstrator implementations. In 5(a), a Bluetooth pairing is established between a Symbian 8.0 camera phone and a Linux laptop with an LED (illustrating, e.g., a wireless access point). In 5(b), two phones are paired using the display of one phone as the bi-state light.

(a) Pairing phone and laptop



(b) Pairing two phones

Figure 5: Scenarios for the proof-of-concept implementation

Our algorithm makes bit reception quite tolerant. The data can be received at a distance of several tens of centimeters, the implementation is agnostic to camera focus problems and tolerates a fair bit of camera shaking, turning, etc. The real-time progress of the matching is indicated at runtime on the handset screen by displaying two parameters: percentage of the string successfully received so far and a related confidence level.

Figure 6 gives a more detailed description of the user interface of our Symbian implementation during pairing with the laptop. In Figure 6(a), the user starts the pairing from a menu.[7] In Figure 6(b), the phone scans the Bluetooth neighborhood and finds the laptop. In Figures 6(c) and 6(d), the phone starts recording with its camera and the user positions the phone so that the blinking of the LED is shown in the viewfinder. The recording status is updated in the viewfinder in real-time. In 6(e), the pairing is complete for the phone once the correct checksum has been received and accepted. The success is reported to the user, who is instructed to accept the pairing at the access point to achieve mutual authentication.

With our setup, a 24-bit checksum signaled (1 error accepted) with the laptop is received and matched by the camera phone. The execution times for a positive indication (match) is typically in the range of 5 to 8 seconds. The increased execution time is the price we pay for achieving

[7]The pairing must be initiated also from the laptop side. The rationale for this is explained in Section 5.2.

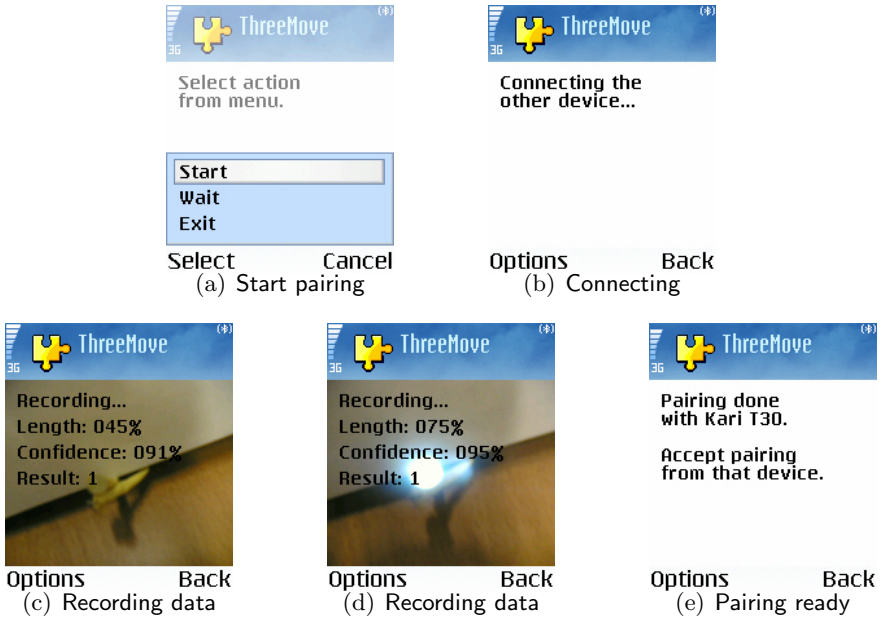


Figure 6: Screen-shots from the Symbian implementation

visual channel authentication with devices that can not afford a full display. As mentioned, we consider these parameters acceptable for ordinary home use. A more secure version (32-bit checksum with 1 error) ranges from 8 to about 15 seconds.

4.3 Extending the Bandwidth on Better Displays

As we saw in Section 4.2, using VICsh with a single light source, and limiting the attack success probability to 2^{-20}, the execution time cannot be smaller than about 5 seconds.

A natural question is whether any speedup in the execution time is possible if there were multiple light sources or in other words, a better display. In this section, we describe the design and analysis of a new *video codec* that can be used to set up a visual channel between a device with a small display and a device with a video camera. Our motivation was to investigate two different questions: whether the video codec can significantly improve the transfer time of a short checksum (15-20 bits), so that it can be used to reduce the execution time of secure pairing, and whether the video codec can enable applications other than secure pairing. In the remaining section, we discuss that even with straight-forward and naive techniques, such a video codec can be designed and that it performs reasonably efficiently.

Encoding Process. The idea of the encoding process is to represent the bits of data to be transmitted after encoding it for error correction into slots (rectangles) of black and white colors (say, 'black' to encode bit '0' and 'white' to encode bit '1') displayed in the form of animated frames at a certain rate. The number of slots that can be displayed in one frame depends upon the size of the display on the device and the video capturing ability of the decoding device, and the number of such frames depends upon the amount of data to be transmitted (e.g., 20 bits plus some error correcting bits in case of the pairing application) and the display rate. Following are the three main constituents of the encoding procedure:

1. Beacon Slot: To allow the decoding device to be able to capture all the frames, the display rate R_d at the encoder should always be smaller than the capture rate R_c at the decoding device. However, since $R_c > R_d$, the decoding device captures more frames than displayed by

the encoding device, some of which get repeated a number of times. In order for the decoder to be able to identify and discard these repeated frames, we devote one slot in each frame (say, at the top left corner) for a beacon frame, which always blinks, i.e, its color always changes from black to white, white to black, and so on. If the decoding device detects that the color of the beacon slot in the current frame is the same as the color of the beacon slot in the previous frame, it can safely discard the current frame. We observe through experiments that $R_d = 10$ frames/second is a reasonable display rate for most camera phones for which $R_c = 15$ frames/second.

2. Blinking Corners: Moreover, to facilitate the decoding device in detecting the screen of the encoding device in the captured frames, we use a small number of always-blinking pixels at the corners of the displayed frames at the encoding device. This simple technique allows the decoder to efficiently identify the exact location of the screen, as we illustrate in the decoding process part below.

3. Marker Frames: In Section 4.2, we exploited the fact that in the case of secure device pairing, the receiving device knows what string to expect via the visual channel. But since we want this video codec to be potentially usable in other applications, we cannot make this assumption. Therefore, since the decoding device may start recording at any bit position in the data string, we need to transmit the data frames at least twice, separated by a small number of marker frames.

Decoding Process. The decoding process involves capturing the video frames displayed by the transmitter, and using these frames to first detect the location of the screen, and then to read the data bits. We describe each of these procedures separately as follows:

1. Locating the Screen: We use the blinking corners (as described previously) in the display to locate the screen. The algorithm is very simple: on input of a certain number t of consecutive frames, denoted by $F_1, \cdots, F_t$, compute a "Sum-of-Differences" frame SD, such that $SD = \sum_{i=2}^{n} |F_i - F_{i-1}|$, and scale SD to pixel values 0 to 255, to obtain an image F. Note that the $|F_i - F_{i-1}|$ denotes the image corresponding to the absolute difference between the pixel values of F_i and F_{i-1}, and adding two images means adding their corresponding pixel values. Notice that the image F brightens the always changing or blinking pixels values (such as the ones corresponding to the corner pixels and the beacon slot) and at the same time darkens the ones which hardly change. See Figure 7(b) for an example F image. Now, to further brighten the smaller regions corresponding to the corner pixels and to darken the other bigger bright regions (such as the one corresponding to the beacon slot), we use a standard tool in image processing called *convolution product*.

Figure 7(c) shows the convoluted image that has only the corner pixels bright. Once the convoluted image is obtained, it is easy to retrieve the corner pixels by using thresholding technique (e.g., in Figure 7(c), all pixels with values greater than 210 correspond to the corner pixels) and thus to locate the screen of the encoding device. Through experiments, we notice that the above algorithm performs quite robustly to detect the screen location if $t = 10$, i.e., if it is given 10 frames as input. The algorithm is also robust to rotation of the screen.

2. Reading the Data: Whenever two corresponding data slots in two consecutive frames have the same color, i.e., both black or both white, we obtain a black or white slots, respectively. However, when they have different colors, we obtain grey slots. The decoder first determines the color of each slot by looking at the distribution of pixel values in the slot and using simple thresholding technique. For example, if maximum number of pixels have values in range $(0, 85)$, the color is black, if they have values in range $(190, 255)$, the color is white, otherwise the color is grey.

Now, if the color of the beacon slot B_i in current frame is the same as the color of the beacon slot B_{i-1} in the previous frame, the current frame can be safely discarded. If the color of (non-beacon) slot S_i is black, output data bit as 0; if it is white, output 1; otherwise if the color is grey, output the complement of the output of S_{i-1}.

(a) A Few Captured Frames

(b) Sum-of-Differences Image

(c) Final Convoluted Image

Figure 7: An example of various images in the decoding process (with 4 slots per frame)

Error Correction. Since we aim for applications besides pairing, we also need to use a robust error correction scheme for the video channel. Currently we use Reed-Solomon (RS) forward error correcting codes [14]. Reed-Solomon is one of the strongest error correcting codes known today, and applies very neatly in our scenario. Firstly, we have observed through experiments that we get errors in bursts (for example, errors in all the slots of one whole frame). Since, the RS codes operate on and corrects errors in symbols of a certain number of bits, they are well-suited to our codec. Secondly, RS codes are capable of correcting errors both in cases of erasures (such errors occur when it is known which symbols are corrupted) and non-erasures. In our codec, we get errors of both types. Erasures occur when it is very difficult to determine the exact color of a particular slot using the method of thresholding as described in the previously (for example, when the maximum number of pixel values are distributed around the boundaries of the thresholds for black and grey or grey and white). Non-erasures occur when we get errors, but we can't predict their locations.

With the (k, n) RS error correction with m-bit symbols, where $n = 2^m - 1$, if there are e erasures and s non-erasure symbols in the received data, the RS code is capable of correcting them as long as $e+2s \leq n-k$. For example, to send out 20-bits of data in the pairing application, we can use $(8, 4)$ RS codes (which is shortened from RS code $(31, 27)$), which corrects e erasures and s non-erasures in 5-bit symbols if $e + 2s \leq 4$.

Implementation and Timings. We have implemented our preliminary video codec prototype using Python Imaging Library[8] on Linux. In the current implementation, our decoding algorithm is given as input the video frames captured from a camera phone. Here, we report on some timing results based on the initial testing that we have done with this Python codec.

To send out 20-bits of original data (or 40-bits of encoded data) with $(8, 4)$ Reed-Solomon codes, as described in before, with 10 frames/second display rate, it takes around 3 seconds, when the display size is capable of displaying only 4 slots a frame, and almost 1 second when the display consists of 8 slots per frame. The screen location detection algorithm takes $2 - 3$ seconds with 10 frames on input, and the decoding and correcting of the data takes almost a second. Overall, it takes approximately $5 - 7$ seconds for the whole process.

These timing results are only preliminary. We anticipate the performance to improve when the python implementation is ported to a native C++ implementation on the Symbian platform.

[8] http://www.pythonware.com/products/pil/

Yet, it is not clear if the execution time for transferring a short integrity checksum can be significantly reduced.

5 Discussion

In this section we discuss the applicability of our results, examine practical use cases, discuss related issues like performance, device discovery, and usability and briefly mention other related work.

5.1 Comparison of Different Protocols

Table 3 summarizes our recommendations on how mutual authentication can be achieved with different device type combinations. If both devices have camera and display, mutual authentication can be achieved either using SiB or VIC. SiB can be used with camera-only devices which can have static barcodes affixed to them. The case of two display-only devices is out of scope for this paper, and the basic MANA techniques which require the user to visually compare two short strings [5, 6] can be used. In all the other cases, VIC could be the best choice since it provides mutual authentication and potentially better usability.

Y has $\rightarrow$ X has $\downarrow$	Camera and display	Camera only	Display only
Camera and Display	SiB/VIC	VIC	VIC
Camera only	VIC	SiB[a]	VIC
Display only	VIC	VIC	MANA

[a] Both devices need static barcode labels affixed to them.

Table 3: Recommended protocol to achieve mutual authentication for given device type combinations

Table 4 summarizes when to use the two different flavours of VIC: If either one of the devices has a full display, then plain VIC as described in Section 3 can be used. Otherwise VIC combined with MA-3 (which we called VICsh) can be used. Table 4 also summarizes the execution time measurements for the two cases. The execution times for the constrained display case or for the limited display is substantially longer than in full display case. Despite this, we stress that this case is extremely relevant, since not all devices have full displays to support the display of barcodes. Commodity devices like access points are very cost sensitive and it is highly unlikely that full displays are added to such devices. In addition, devices like headsets are so small that adding full displays is not possible. Also, note that the timings for constrained display case are geared for home usage scenarios.

Since the bandwidth requirement for VICsh protocol is low, this protocol could be used in scenarios where it is not possible to reach the bandwidth required by the VIC protocol. One example of such a scenario is a WLAN access point that is mounted high up on the wall or ceiling. It is not possible to read the barcode affixed to such an access point with the current camera phones, but it might be possible to read the "blinking" of the access point if the light source is powerful enough.

The preliminary timing results for transferring short strings using the video codec described in Section 4.3 are more or less comparable to the timing results for the single light-source

Display type	Recorder type	Protocol	Execution time
Full display	Still camera	VIC	1 second[a]
Limited display	Video camera	VICsh	5-7 seconds[b]
Constrained display	Video camera[c]	VICsh	5-8 seconds[d]

[a] Symbian OS implementation on Nokia 6600 [12]
[b] Python implementation on PC
[c] Can also be a light sensor
[d] Symbian OS implementation on Nokia 6630

Table 4: Applicability of different flavors of VIC

approach described in Section 4.2. The former approach is more robust because of the forward error correction. The latter approach is somewhat cheaper. For ordinary uses of secure pairing, the single light-source approach may be more suitable even when the available display is slightly better than a single light-source. However, the video codec is a useful service for device discovery applications (as we discuss next) where several hundred bits of information need to be transferred via the visual channel, and there is no other communication channel between the two devices. For example, a small section of the television display may be used to transmit the address of a web page relating to the television program in progress. Note that using a sequence of barcodes to encode this information is not a viable option since it would require the user to capture the barcode, notice when the barcode changes and ensure that each barcode is recorded.

5.2 Device Discovery Strategies

Previous proposals on security initialization using out-of-band methods [17, 2] have argued that one of the main benefits of using an out-of-band channel for security initialization is the fact that device discovery is part of the OOB message exchange. For example in the approach proposed by Balfanz et al. [2] the devices exchange complete addresses over infrared, and thus no in-band device discovery is needed. In SiB approach, the device discovery is done manually (because current phones can not display big enough bar codes to contain both the address and the hash of a public key), but the authors state that the optimal solution would be to encode both the address and the public key hash to the bar code.

We argue that in many scenarios an in-band device discovery is actually needed before the OOB message exchange. The increasing number of different OOB channels (such as infrared, camera and full display, camera and single LED etc.) results in situations where the user might not always know which OOB to use with the two particular devices at hand. For example a user wanting to pair a camera phone (camera, display, no infrared) with a laptop (infared, display, no camera) might be confused about the different OOB possibilities. It should not be the user's burden to figure out which OOB to use (and how), but instead an in-band device discovery should take place and the best mutually supported OOB channel should be negotiated in-band and the user should be guided to use this OOB. Negotiations must be protected against bidding-down attacks in the usual manner, by having the parties exchange authenticated confirmations of the negotiation messages once key establishment is completed (as is done with the "Finished" message in TLS[4]). As long as the chosen authentication mechanism can not be broken in real-time, attempts to bid-down will be detected by this check.

In order to conveniently discover the desired device in-band, the user must put one of the devices into a temporary special discoverable mode so that the user does not have to select the correct device from a long list of (probably meaningless) device names. We call this action *user conditioning*. From the user's point of view this action can be performed, e.g., by pressing a

button on the device or by selecting a menu option.

Not all bearers support in-band discovery without manual device selection. Likewise, pure out-of-band discovery is not always feasible with constrained OOB channels. In these cases, the constrained OOB can be used to improve the usability of the in-band discovery process. A device can, e.g., send the last 10 bits of its address over OOB. At the same time the other device can scan and automatically discard devices whose address does not match these 10 bits. With high probability the correct device can be selected automatically and the user does not have to be presented a list of device names.

5.3 Usability Considerations

The security of VIC and VICsh relies on the user answering affirmatively in the last step (in Figures 2 and 3). If device B rejects the key agreement and indicates failure to the user, but the user inadvertently answers affirmatively in the last step, device A would conclude that the key agreement was authenticated even though B does not. One way to mitigate the impact of this failure is as follows. A picks a secret k and sends it via the visual channel, along with the checksum. If B accepts the key agreement, it can use the resulting secure channel to prove knowledge of k. A will accept the key agreement only if the user accepts in the last step, *as well as* a proof of knowledge k is received via the secure channel. On the downside, the additional check increases the amount of data transferred over the visual channel, thereby increasing the execution time. Moreover, the additional check is effective only if the attacker can not snoop on the visual channel.

Another way to reduce the likelihood of accidental (or out of habit) confirmation is to use a specific confirmation button only for the purpose of secure device pairing. The downside is the cost of adding such a button.

Whether this accidental confirmation is a real concern can only be determined by extensive usability testing. To date, none of the research papers dealing with the problem of secure device pairing have reported substantial *comparative* usability testing. The only exception is [1], which presents some usability analysis of their approach of using infrared as an OOB channel. Given the level of recent interest in this area which has resulted in several pairing approaches, a comprehensive comparative usability testing will be a very valuable research contribution. We are addressing this in our current work.

5.4 Denial-of-Service

Another concern is the possibility of a denial-of-service attack. An attacker can disrupt a pairing attempt between two devices by simultaneously initiating pairing with one or both of the same devices. Accidental simultaneous pairing is likely to be very rare because of the user conditioning described in Section 5.2. Thus, if a device detects multiple pairing attempts, the best strategy may be to ask the user to try again later, rather than ask the user to choose the correct device. Moreover, sending part of the device identifier via the visual channel, as described in Section 5.2, will help in picking the correct device in case of multiple parallel device pairing attempts. Note that in wireless networks, elaborate attempts to protect the pairing protocol against malicious attempts of denial-of-service are not cost effective because an attacker can always mount denial-of-service by simply disrupting the radio channel.

5.5 Other Related Work

Recently Goodrich, et al. [7], proposed a pairing mechanism making use of audio as the OOB channel. Their idea is to encode the public key hash value into an auditorially-robust, grammatically-correct sentence, which is displayed on one device and read out on the other using a voice synthesizer. The user then manually compares the two versions of the sentence in

order to authenticate the public key. However, this scheme also suffers from the same problems as does SiB: namely, mutual authentication is either not possible (i.e, when one of the devices does not have an audio output and a display), or could be quite inefficient and taxing on the users. Fortunately, they can also use the MANA family of authentication protocols similar to our proposal in Section 4.1.

6 Conclusions

In this paper, we proposed several improvements and extensions to the recently proposed approach of using a visual channel to implement secure pairing. We showed how strong mutual authentication can be achieved using just a unidirectional visual channel, which could also improve the usability of the pairing process.

We then showed how visual channel authentication can be used even on devices that have very limited displaying capabilities, such as a single LED. Commoditized devices like wireless access points, and devices with form factor limitations like headsets, cannot afford to have full displays capable of displaying barcodes. Our contribution makes it possible to use visual channel authentication even on such devices.

It would be feasible to trim down the camera to a simple light sensor. Although at first glance this might seem to be the same as a one-way infrared communication channel, there are important differences in terms of user perception and cost: first, a user can easily see a light source, and can detect the presence of a false source; second, adding an infrared interface for the purpose of secure device pairing is not an economically viable option for commodity devices like wireless access points or Bluetooth headsets; but typically they tend to have one or more LEDs which can be used to implement the technique we propose. By integrating a flashing light-source on one device and a light sensor on another, two wireless sensor devices can thus be efficiently paired.

Finally, we proposed a *video-based* codec which may help improve the speed of secure pairing in devices with less constrained, but not full, displays, as well as may lead to applications other than secure device pairing.

Acknowledgements

We are grateful to Adrian Perrig, who shepherded the conference version of this paper, Jonathan McCune, Markku Kylänpää, and the anonymous reviewers for their thoughtful and constructive feedback which helped us improve the paper. We thank Jonathan also for giving us the source code for SiB. We also thank Marie Selenius, Dr. Niklas Ahlgren, and Dr. Valtteri Niemi for insights into the counting argument, Kaisa Nyberg and Stanisław Jarecki for valuable feedback on our protocols, and Aurélien Francillon for several discussions regarding our video codec.

References

[1] Dirk Balfanz, Glenn Durfee, Rebecca E. Grinter, Diana K. Smetters, and Paul Stewart. Network-in-a-box: How to set up a secure wireless network in under a minute. In *USENIX Security Symposium*, pages 207–222, 2004.

[2] Dirk Balfanz, Diana Smetters, Paul Stewart, and H. Chi Wong. Talking to strangers: Authentication in ad-hoc wireless networks. In *Network and Distributed System Security Symposium*. The Internet Society, 2002.

[3] RVSI Acuity CiMatrix. Data Matric Barcodes, 2005. Available at http://www.rvsi.net/.

[4] Tim Dierks and Christopher Allen. The TLS protocol version 1.9. Internet Engineering Task Force, RFC 2246, January 1999.

[5] Christian Gehrmann et al. SHAMAN Deliverable: Detailed Technical Specification of Mobile Terminal System Security, May 2002. Available at www.isrc.rhul.ac.uk/shaman/docs/d10v1.pdf.

[6] Christian Gehrmann, Chris J. Mitchell, and Kaisa Nyberg. Manual authentication for wireless devices. *RSA CryptoBytes*, 7(1):29 – 37, Spring 2004.

[7] Michael T. Goodrich, Michael Sirivianos, John Solis, Gene Tsudik, and Ersin Uzun. Loud and Clear: Human-Verifiable Authentication Based on Audio. In *International Conference on Distributed Computing Systems (ICDCS)*, July 2006. Available at `http://www.ics.uci.edu/ccsp/lac`.

[8] Stephen R. Hanna. Configuring Security Parameters in Small Devices, July 2002. draft-hanna-zeroconf-seccfg-00.

[9] Jaap-Henk Hoepman. The ephemeral pairing problem. In *Proc. Int. Conf. Financial Cryptography*, number 3110 in Lecture Notes in Computer Science, pages 212–226. Springer, 2004.

[10] Sven Laur, N. Asokan, and Kaisa Nyberg. Efficient mutual data authentication based on short authenticated strings. IACR Cryptology ePrint Archive: Report 2005/424 available at `http://eprint.iacr.org/2005/424`, November 2005.

[11] Anil Madhavapeddy, David Scott, Richard Sharp, and Eben Upton. Using camera-phones to enhance human-computer interaction. In *Sixth International Conference on Ubiquitous Computing (Adjunct Proceedings: Demos)*, 2004.

[12] Jonathan M. McCune, Adrian Perrig, and Michael K. Reiter. Seeing-is-believing: Using camera phones for human-verifiable authentication. In *Proc. 2005 IEEE Symposium on Security and Privacy*, pages 110–124. IEEE, 2005.

[13] Harry Nyquist. Certain topics in telegraph transmission theory. *Transasctions of the American Institute of Electrical Engineers (AIEE)*, 47:617–644, 1928.

[14] Irving S. Reeed and Gustave Solomon. Polynomial codes over certain finite fields. In *Journal of Society for Industrial and Applied Mathematics*, 1960.

[15] Michael Rohs and Beat Gfeller. Using camera-equipped mobile phones for interacting with real-world objects. In Alois Ferscha, Horst Hoertner, and Gabriele Kotsis, editors, *Advances in Pervasive Computing*, pages 265–271, Vienna, Austria, April 2004. Austrian Computer Society (OCG).

[16] Nitesh Saxena, Jan-Erik Ekberg, Kari Kostiainen, and N. Asokan. Secure device pairing based on a visual channel. In *IEEE Symposium on Security and Privacy (ISP'06), to appear as short paper*, May 2006.

[17] Frank Stajano and Ross J. Anderson. The resurrecting duckling: Security issues for ad-hoc wireless networks. In Bruce Christianson, Bruno Crispo, James A. Malcolm, and Michael Roe, editors, *Security Protocols Workshop*, volume 1796 of *Lecture Notes in Computer Science*, pages 172–194. Springer, 1999.

[18] Serge Vaudenay. Secure communications over insecure channels based on short authenticated strings. In *Advances in Cryptology - CRYPTO 2005*, number 3621 in Lecture Notes in Computer Science, pages 309 – 326. Springer Verlag, 2005.

[19] Simon Woodside. Read real-world hyperlinks with a camera phone, February 2005. Available at http://semacode.org.