Rainbow Crypt: Securing Communication through a Protected Visual Channel

Alberto Bartoli, Eric Medvet, Giorgio Davanzo
DI^3 - Università degli Studi di Trieste
Trieste, Italy
Email: bartoli.alberto@units.it

***Abstract*—Electronic devices capable of wireless communication are becoming ubiquitous. They enable a wide range of novel applications but these are often difficult to deploy in practice because wireless channels provide ample opportunities to attackers. A number of approaches have been proposed for building secure channels in these scenarios. An approach that would be simple, general and effective consists in establishing a shared secret between two devices by placing them in physical contact for a few seconds. This approach has not been exploited in practice due to the lack of common interfaces. We demonstrate the practical feasibility of the approach for devices equipped with a small LCD screen and cameras. We transfer secret keys between Android-based smartphones put in contact with each other for just a few seconds. The transfer occurs across a visual channel that cannot be intercepted.**

***Keywords*-Secure device pairing; wireless key exchange; Seeing is Believing; out-of-band; OOB; camera phones.**

I. Introduction

Electronic devices capable of short-range wireless communication, such as WiFi, Bluetooth and Near Field Communication (NFC), are becoming ubiquitous, which enables a huge diffusion of a variety of applications requiring secure communication amongst these devices. Access control, electronic payment, monitoring of medical devices, home automation are just a few examples. Since wireless channels are easy to eavesdrop and messages can be dropped and replayed easily, however, the required security properties cannot be obtained automatically. What makes the problem difficult is the lack of a global infrastructure for establishing preconfigured secrets amongst devices. Consequently, a number of approaches have been proposed for implementing forms of secure device pairing, i.e, for bootstrapping secure communication between two human-operated and resource-constrained devices over a short-range wireless channel. Recent efforts advocate the use of an auxiliary channel where the users operating the devices may perform authentication and establish shared secrets: once the devices have been paired on this out-of-band (OOB) channel, they may communicate through the wireless channel securely.

An audio OOB channel is proposed in [1], where a device public key is transformed into a syntactically correct English-like sentence and read aloud by a Text-to-Speech engine. The user is expected to validate the sentence, for example to make sure that two devices to be paired indeed utter the same sentence. The same approach may be used with devices that do not have a speaker but have a display, i.e., an OOB visual channel. A similar approach is proposed in [2], where short messages are displayed on the user terminals and manually authenticated. A visual OOB channel based on 2-D bar codes and camera phones is proposed in [3], along with protocols for solving several security-related problems in device pairing. An improvement to this approach is proposed in [4] that relaxes the need for a fully bidirectional OOB visual channel, requiring instead only a "single-bit" channel in one of the two directions, e.g., a blinking led.

A very simple solution was proposed in a seminal paper [5], that advocated the mere use of *physical contact* as a general and effective mechanism for secure device pairing. The bits constituting the shared secret are transferred through an electrical contact between the devices to be paired and the transfer may occur in plaintext. Mutual authentication is provided by the user that decides to put the devices in contact (the obvious implicit assumption being that each device is indeed what the user assumes it is). In a sense, physical contact between the devices to be paired constitutes a form of OOB channel that provides mutual authentication and secrecy natively. Despite its simplicity, this approach did not succeed in practice due to the lack of common interfaces for establishing the physical contact amongst devices.

In this work we investigate the possibility of actually implementing this approach in modern devices equipped with a small color screen and a camera—smartphones being just an example of such devices. The potential advantages in terms of speed and simplicity are evident: by merely placing the two devices in contact for a few seconds, i.e., with the camera of one placed above the screen of the other, the devices can establish a shared secret. This secret can then be used over the wireless channel to obtain privacy, mutual authentication and alike. Attacks on the OOB visual channel used for establishing the shared secret would be extremely difficult to carry out, as the images on the screen would be physically shrouded by the other device. Implementing this simple idea in practice is difficult, however, in particular because cameras are not designed to focus on very short distances. Our contribution consists in showing that the corresponding problems can be solved and the approach is indeed practical. We experimented with low-end Android devices and succeeded in transferring 42-bit keys in around

978-1-4577-1676-8/11/$26.00

5 seconds. The actual transmission consists of frames with an uniform color. We called our system *Rainbow Crypt*—perhaps with some abuse as we use a set of 4 possible colors.

A variety of security protocols and interesting applications may be deployed in practice based on the strong security properties of the resulting OOB visual channel. For example, a mobile phone could be used as enabling device in an access control system. Upon entering, the user would place his phone next to the screen of a control system (or maybe even the phone of the warden) for a few seconds. Then, the user would be authenticated in the various locations by advertising his presence to a WiFi infrastructure, which would do so securely based on the shared secret established upon entrance. As another example, a nurse could place the camera of his phone next to the screen of a patient monitoring device. The readings of the device could be streamed securely to the nurse device across a WiFi infrastructure. One could devise secure payment schemes based on mobile phones. Instead of swiping a chip-and-pin card in the selling machine, the user would place his mobile device camera on the selling machine LCD screen, receiving a shared secret to be used in the actual transaction over a bluetooth or NFC channel. Indeed, a mobile phone would be able to execute in the transaction more complex operations and protocols than a chip-and-pin card. The approach could also complement traditional existing payment schemes, for example, as a defense to attacks that obtain ATM pin codes simply by means of observation or hidden spy-cameras [6], [7]. Users could transfer pin codes by coupling their phones to the ATM surveillance camera (assuming that attackers cannot mount fake surveillance cameras that would go unnoticed by users; of course, the assumption is that users are able to authenticate the devices to be paired).

II. Our Approach

We consider the establishment of a secure and private communication channel between two resource-constrained devices. The transmitter and receiver are equipped, respectively, with a small color screen (an LCD screen in our prototype) and a camera. Key feature of our approach is that transmitter and receiver are placed in direct contact with each other. This setting makes communication difficult, as the receiver camera cannot properly focus on the transmitter screen. On the other hand, the visual information transmitted on the channel is virtually impossible to intercept. The LCD area that transmits the message may be kept as small as the camera lens itself—normally 1/3" is enough—thus placing the camera device on the LCD screen will shield the emitting surface completely.

Our experiments, discussed in the next sections, show that the above communication channel is indeed severely limited in bandwidth, but nonetheless it can be used to securely transfer sensitive informations, like a pin code or a shared key to be used on a wider bandwidth channel.

Table I
Devices used in the experiments

Sender (display)	Receiver (camera)
HTC Nexus One	HTC Nexus One
HTC Tattoo	HTC Tattoo
HTC Magic	HTC Magic
Dell 21" LCD	Logitech Webcam
Sun 19" LCD	
MacBoock Pro	

III. Experiments and Results

We experimented with several devices, listed in Table I. The list includes Android smartphones ranging from low-end to high-end models, traditional LCD displays and a PC webcam.

We developed a prototype for the Android platform, and tested it on the devices listed in the table. The prototype consists of a transmitter and a receiver. When testing the coupling between LCD screens and webcam, we ran the application within the official Android emulator running on a PC emulator, i.e., one endpoint transmitting on the PC screen and the other endpoint receiving from the webcam.

A. Bits per frame

It is reasonable to assume that nowadays even the cheapest mobile phones are equipped with cameras able to take 24-bit colorful pictures. However, these cameras are designed to operate on short-medium distances, i.e., from 2 feet up[8]. Our approach requires instead a near-zero distance between camera and subject, which introduces several technical challenges.

First and foremost, is is impossible to use the transmitter display to show more than a single color. Since the focusing can not be achieved on such a short distance, everything more complicated than a uniform color image would appear as a single blurred color anyhow. This fundamental constraint introduces severe limitations on frame content and, consequently, on the channel bandwidth. In this section we will elaborate on the number of colors that can be reliably detected per frame.

In addition to the focusing issue, there are two more factors that affect the camera output and are crucial in our context: *white balancing* and *exposure*.

The phone camera white balancing software attempts to average the image color to a neutral grey. When the image is composed of a single color—as we are forced to do as discussed above—the resulting image output by the camera is a distorted color, often unpredictable as it depends heavily on the settings and type of the devices involved. On all the cameras we tested, though, this functionality can be disabled easily, in software.

On the other hand, coping with exposure is more difficult. The phone camera software always tries to select an exposure interval for the camera sensor such that the resulting

image outputs a standard average luminosity. For example, a dark grey and a light grey are transformed to a medium grey. As it turns out, this functionality makes different color tonalities impossible to distinguish, i.e., transmitting a certain color tonality may result in receiving a widely different tonality.

We investigated this problem with a number of dedicated experiments. In particular, we transmitted frames of uniform color, taken from a set of C colors from the RGB palette and chosen as described below. We figured our that the maximum number of colors that every camera can discern is $C = 4$.

We also experimented on which colors were to be selected. We discovered that as long as the RGB components were distant from each other, the actual colors did not matter. In the remaining experiments we used the following colors: Red (255,0,0), Blue (0,0,255), Violet (255,0,255) and Black (0,0,0).

To recognize the color of a frame, we proceed as follows:

1) select P pixels having the lowest Manhattan distance from the image center,
2) convert the YUV values output by the phone camera software to RGB color space,
3) compute the average color $(c_{\mathrm{red}}, c_{\mathrm{green}}, c_{\mathrm{blue}})$
4) apply a threshold $\tau_{\mathrm{component}}$ to each color component: if $c_{\mathrm{component}} > \tau_{\mathrm{component}}$ we set it to 255, otherwise we set it to 0.

After some experimentation, we found that the optimal threshold values are $\tau_{\mathrm{red}} = 148$ and $\tau_{\mathrm{blue}} = 180$. Since we did not use the green component, we did not research the relative threshold.

Since the YUV-RGB conversion introduces a relatively high computation load on the device, we chose a small P value in order to limit this load. When investigating the effect of increasing P on computation time, however, we discovered that using a small value for P is essential for improving the color recognition ability of the receiver. The reason is due to the effect that the camera lens has on the zero-distance configuration: the color is more truthful in the image center, while it becomes lighter on the edges. For this reason, we decided to use $P = 20$ in all the following experiments.

B. Frames per second

Having determined the number of colors—i.e., number of different symbols—that can be used, the next step is determining the transmission rate, i.e., the number of frames per second that can be transmitted reliably. The cheapest mobile phone amongst those that we tested can capture videos at 21fps. We decided to experiment with a frame rate bounded at 10 fps,according to the Nyquist-Shannon sampling theorem[9]. These experiments provided unacceptably bad performance, though.

We investigated this scenario in detail and discovered two factors that forced us to use a significantly lower frame rate: exposure and platform limitations.

Exposure, as expected, varies during the transmission; occasionally, when the display shows a black picture for a prolonged time, the exposure drops below 1/FPS, thus making it impossible for the device to record pictures at the desired rate. We observed that usually the longest exposure is 1/10s; hence, setting $\mathrm{FPS} \leq 10$ should resolve the exposure issue. Please remember from Section III-A that the exposure value can not be changed or set.

Concerning the *platform limitations*, we discovered that although Android allows video applications to select the desired frame rate, in practice there is a significant variation in the frame sampling intervals. For instance, if an application specifies that images have to be taken at 20 fps, the device does not capture an image every 50 ms—some frames may be 200 ms apart, some others may differ of just a few ms. That is, Android guarantees the desired frame rate over long periods, but the duration of the sampling interval is not constant. It follows that if the transmitter is implicitly clocked at a certain frequency, on the receiver side there may be unpredictable gaps, frame duplication and so on.

In order to circumvent these issues, we experimented with several combinations of transmission rate and sampling frequency on the receiver side. We found the best values when the transmitter sends data ad 5fps (corresponding to 10bit/s) and the receiver captures images at 20fps. We also had to experiment with several different methods in order to cope with the Android frame sampling issues, as described in the next section.

C. Sampling method

We experimented with several methods for sampling the received frames.

We define the *first symbol period median* t_0 as the median instant of the first frame. Thus, being $T = 0.2s$ the frame period, the k-th following frame will start at $t_0 + (k - \frac{1}{2})T$ and finish at $t_0 + (k + \frac{1}{2})T$.

For the first three methods hereafter described, we compute t_0 after the first symbol transition.

- **Nearest Method** is the simplest one; every time a frame is available from the camera we associate it with a timestamp. Then, we select those frames whose timestamps are closer to $t_0 + kT$. All the other frames are discarded, and the symbol is extracted from the selected frames.
- **Mean Method** all the frames are grouped basing on the closer $t_0 + kT$, and the YUV values provided to the color recognizer are averaged among these groups.
- **Weighted Mean Method** is similar to the previous one, except that YUV values are weighted depending on their squared distance from $t_0 + kT$.
- **Weighted Mean Method with Sync** adds a synchronization sequence at the beginning of the transmission

Table II
SAMPLING METHODS ERRORS

Method	Average (%)	StdDev (%)
Nearest	12.0	23.6
Mean	11.4	20.2
wMean	10.3	20.2
wMean Sync	0.3	1.0

Table III
PAYLOAD FOR DIFFERENT TRANSMISSION LENGHTS

Transmission Time (s)	Payload (bits)
1	2
2	12
3	22
4	32
5	42

as described in Section III-D in order to better estimate t_0.

The results are presented in Table II in terms of average error and standard deviation; we assessed every method using 20 sequences of variable lenghts on different devices.

It can be noted that while the first three methods behave similarly with an average error of roughly 11%, the introduction of the sync mechanism lowers the average error to a much better 0.3%.

Table III shows the net payload achieved for different transmission lenghts when the system is working with the "Weighted Mean Sync" method. Hence, a coupling time of 5-10 seconds will suffice to establish shared secrets useful for practical applications.

D. Synchronization algorithm

The synchronization aims at assessing the first period median t_0.

The algorithm requires a synchronization sequence known both to the sender and the receiver. This sequence is composed by two alternating symbols (red and blue) repeated twice, thus requiring a total of 4 symbols.

On the *sender* side, the algorithm simply consists in transmitting the synchronization sequence.

On the *receiver* side, the algorithm is the following:

1) the device starts capturing symbols, associating each frame to a timestamp
2) when the receiver observes the n-th symbol change ($n = 1, \ldots, 3$), it uses the timestamps t_p of the previous frame and t_f of the following frame to estimate the transition instant
$$\hat{t_n} = \frac{t_p + t_f}{2}$$
3) after observing the sync sequence, the receiver computes the first symbol period median t_0 as follows:
$$t_0 = \hat{t}_2 + \frac{\hat{t}_3 - \hat{t}_1}{2} + \frac{7}{2}T$$
4) if the receiver can not observe the sync sequence, it raises an exception.

IV. CONCLUDING REMARKS

The presented system, if correctly used, provides a communication channel that can not be intercepted by human beings or sniffing devices.

The channel bandwidth is limited by technology and platform constraints; however, it is still wide enough to allow the safe and fast transfer of a shared key—especially if the shared key is considered valid for a limited amount of time.

We did not explore the usage of an error control (or correction) code for the following reasons: (i) the error rate is quite low (0.3%) and (ii) even if there is a transmission error, the receiver would discover it when trying to authenticate or communicate with the sender device.

ACKNOWLEDGMENT

The authors would like to thank Giacomo Petronio for his hard work during the initial stage of this research.

REFERENCES

[1] 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*, 2006, p. 1010.

[2] S. Laur and K. Nyberg, "Efficient mutual data authentication using manually authenticated strings," *Cryptology and Network Security*, p. 90107, 2006.

[3] 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*, 2005, p. 110124.

[4] N. Saxena and M. Uddin, "Automated device pairing for asymmetric pairing scenarios," *Information and Communications Security*, p. 311327, 2008.

[5] F. Stajano and R. Anderson, "The resurrecting duckling: Security issues for ad-hoc wireless networks," in *Security Protocols*, 2000, p. 172182.

[6] N. Utakrit, "The phantasm of ATM withdrawal," in *Information Security Management Conference*, 2007, p. 208.

[7] L. Bruce, "Skimming the cash out of your account," *Retrieved July*, vol. 6, p. 2006, 2003.

[8] J. Dave, "Better photos from your camera phone," http://goo.gl/eNLR4. [Online]. Available: http://goo.gl/eNLR4

[9] H. Nyquist, "Certain topics in telegraph transmission theory," *American Institute of Electrical Engineers, Transactions of the*, vol. 47, no. 2, p. 617644, 1928.