Seeing-Is-Believing: Using Camera Phones for Human-Verifiable Authentication *

Jonathan M. McCune Adrian Perrig Michael K. Reiter
Carnegie Mellon University
{jonmccune, perrig, reiter}@cmu.edu

Abstract

Current mechanisms for authenticating communication between devices that share no prior context are inconvenient for ordinary users, without the assistance of a trusted authority. We present and analyze Seeing-Is-Believing, a system that utilizes 2D barcodes and camera-phones to implement a visual channel for authentication and demonstrative identification of devices. We apply this visual channel to several problems in computer security, including authenticated key exchange between devices that share no prior context, establishment of a trusted path for configuration of a TCG-compliant computing platform, and secure device configuration in the context of a smart home.

1. Introduction

Obtaining authenticated values from devices in ways that are easily understandable by non-expert users is currently an open problem. This is best exemplified by the problem a user faces when she wants to securely connect her wireless device to *that* other device (e.g., a network printer, an 802.11 base station, or another wireless device). In general, it is exceedingly difficult to determine which device is at the other end of a wireless connection without out-of-band knowledge. Balfanz et al. describe this as the problem of achieving *demonstrative identification* of communicating devices [4]. We approach this problem with the premise that, in many situations, a user can visually identify the desired device.

*This research was supported in part by National Science Foundation grant number CNS-0433540, U.S. Army Research Office contract number DAAD19-02-1-0389, and by gifts from Bosch and Intel. The views and conclusions contained here are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either express or implied, of ARO, Bosch, Carnegie Mellon University, Intel, NSF, or the U.S. Government or any of its agencies.

As camera-equipped mobile phones rapidly approach ubiquity, these devices become a naturally convenient platform for security applications that can be deployed quickly and easily to millions of users. Today's mobile phones increasingly feature Internet access and come equipped with cameras, high-quality displays, and short-range Bluetooth wireless radios. They are powerful enough to perform secure public key cryptographic operations in under one second.

We propose to use the camera on a mobile phone as a new *visual channel* to achieve demonstrative identification of communicating devices formerly unattainable in an intuitive way. We term this approach Seeing-Is-Believing (SiB). In SiB, one device uses its camera to take a snapshot of a barcode encoding cryptographic material identifying, e.g., the public key of another device. We term this a *visual channel*. Barcodes can be pre-configured and printed on labels attached to devices, or they can be generated on-demand and shown on a device's display.

We apply this visual channel to several problems in computer security. SiB can be used to bootstrap authenticated key exchange between devices that share no prior context, including such devices as mobile phones, wireless access points, and public printers. We use SiB to aid in the establishment of a trusted path for configuration of a TCG-compliant[1] computing platform, and to provide the user with assurance in the integrity of an application running on a TCG-compliant computing platform. We also use SiB to secure device configuration in the context of a smart home.

Outline We survey related work in Section 2 and provide an overview of SiB in Section 3. Section 4 presents the use of SiB for authenticated key exchange between mobile devices. Section 5 explains how to use SiB to achieve demonstrative identification of, and secure connection to, a particular wireless device, with establishment of a

[1]The Trusted Computing Group (TCG) is an organization that promotes open standards to strengthen computing platforms against software-based attacks [2, 3].

trusted path from the user to a TCG-compliant computing platform as a special case. Leveraging a TCG-compliant computing platform, Section 6 shows how SiB can help verify that a particular application currently "owns" the computing platform's display, and provide high assurance that the integrity of that application is maintained. We also show in Section 7 how this technology can be used to achieve slightly weaker—but still quite valuable—security properties in the context of, e.g., a smart home. Our implementation is detailed in Section 8, with a security analysis in Section 9. Section 10 concludes.

2. Related Work

SiB is closely related to work on authentication involving mobile devices, barcode scanning with camera phones, and using mobile phones as trusted intermediaries.

2.1. Authentication

In this section, we study authentication between two co-located entities with no prior trust relationships. Since using a public key infrastructure relies on the existence of trusted certifiers (e.g., hierarchical certification [23] or unstructured certification [52]), we do not consider these approaches here. Similarly, trusted third party approaches, such as Kerberos [31, 42], assume an online trusted authority which may not exist in our setting.

A common mechanism to establish a secure channel between two entities is to use Diffie-Hellman key establishment [12]. Unfortunately, a man-in-the-middle (MITM) attack is possible if the two entities do not share any established trusted information. Bellovin and Merrit propose the encrypted key exchange (EKE) protocol, which prevents the MITM attack if both parties share a secret password [6]. Several researchers have refined this approach [6, 7, 8, 28, 51], but they all require a shared secret password between the two entities, which may be cumbersome to establish in many mobile settings.

Another approach to defeat the MITM attack is to use a secondary channel to verify that the same key is shared by two parties. An approach that several researchers have considered is that a human can manually verify that the generated keys are identical. Since comparing the hashes of the two keys is cumbersome, researchers have devised visual metaphors that represent the hash to make it easier for people to perform the comparison: Levien and Goldberg devise the Snowflake mechanism [16, 27], Perrig and Song propose to use Random Art as a visual hash [32], and Dohrmann and Ellison propose a colorful flag representing a hash of the key [13]. Though these schemes make key comparison easier for the user, they still rely on the user to diligently compare the resulting visual key representations. With SiB, visual device identification is an integral part of establishing a connection between devices, though in a far less overt way.

To defend against MITM attacks, Stajano and Anderson propose to set up keys through a link that is created through physical contact [41]. However, in many settings, devices may not have interfaces that connect for this purpose, or they may be too bulky to carry around. Balfanz et al. extend this approach to use short-range wireless infrared communication [4]. Of all these approaches, theirs is the most closely related to SiB, and we discuss it in more detail in Section 3.3. Čapkun, Hubaux, and Buttyán have further extended this research direction [48]. They make use of one-hop transitive trust to enable two nodes that have never met to establish a key. SiB could leverage this technique equally well.

Researchers have studied the problem where a user wants assurance that what she sees on the screen of a computer is really the content sent from a trusted provider or displayed by a trusted application. Tygar and Whitten consider Trojan horse web applications that may be able to monitor keystrokes and steal private information [46]. They combat this problem using window personalization. This is best described as a heuristic and does not provide cryptographic security guarantees. Ye and Smith propose *trusted paths* from web browsers to users. Their solution is based on window borders that flash in a particular way, such that malicious content or a malicious application is unable to easily match the correct pattern. This solution provides no cryptographic guarantees, and we believe it requires substantial user diligence. We discuss the use of SiB to address these problems in Section 6.

2.2. Barcode Recognition with Camera Phones

SiB depends on a camera phone having the ability to use its camera to recognize two-dimensional (2D) barcodes. Several projects exist that seek to allow camera-equipped mobile phones to interact with physical objects through the use of 2D barcodes. Rohs and Gfeller develop their own 2D code explicitly for use with mobile phones, emphasizing their ability to be read from electronic screens and printed paper [34]. Woodside develops *semacodes*, which is an implementation of the Data Matrix barcode standard for mobile phones [36, 49]. Woodside considers the primary application of semacodes as containers for a URL which contains information about the physical location where the barcode was installed. Madhavapeddy et al. use SpotCodes to enhance human-computer interaction by using a camera-phone as a pointing and selection device [29]. Researchers working on the

CoolTown project at HP Labs propose tagging electronics around the house with barcodes to be read by camera phones or PDAs so that additional data about the tagged device can be easily retrieved [1].

Hanna considers devices with barcodes affixed to aid in the establishment of security parameters [18]. His work considers a smart home, where a user may want to establish a security context for controlling appliances or other devices in a smart-home. In Hanna's work, the barcode contains a secret which is also stored inside the device. Hanna proposes using this secret to enable the secure transmission of commands to the device from a master controller over an untrusted network. We refer to the security property discussed by Hanna as *presence*, where it is desirable that only users or devices close to some device are able to control it. We discuss the notion of *presence* further in Section 7.

2.3. Mobile Phones as Trusted Intermediaries

Burnside et al. explore the possibility of using cameras to authenticate screen contents on an untrusted computer so that a user can securely interact with a remote trusted computer [9]. They require the user to position a camera such that it does not move relative to the screen being authenticated and then perform a pixel-mapping calibration step. Upon completion of setup, the camera-equipped verifying device must constantly process the screen contents on the untrusted device. They also require the camera-equipped verifying device to communicate with a user-specific proxy. In contrast, SiB is a lightweight solution, capable of running on mobile phones.

Wu et al. propose the use of mobile phones to enable secure web authentication from an untrusted terminal [50]. They utilize a trusted proxy to manage user passwords and authentication data, preventing long-term authenticators (e.g., cookies) from ever touching the untrusted terminal. Gieseke and McLaughlin extend this work, improving communication speed and using a keyed hash to authenticate the security proxy server and the mobile client [15].

Ross et al. develop rule-driven transformation functions that enable access to Internet services from multiple kinds of devices, including mobile phones [35]. They consider rules that enhance security when working from an untrusted kiosk, making it unnecessary for the user to enter sensitive information (such as credit card numbers and mailing addresses). This scheme requires manual configuration for each Internet service and user device type.

SiB enhances security for the user interacting with the computer in front of her, and is complementary to the techniques of Wu et al. and Ross et al.

3. Seeing-Is-Believing (SiB)

With SiB, a mobile phone's integrated camera serves as a visual channel to provide demonstrative identification of the communicating devices to the user. By *demonstrative identification*, we mean the property that the user is sure her device is communicating with *that* other device. In SiB, the user identifies *that* other device visually. This serves to strongly authenticate one or both devices since the user knows precisely which devices are communicating. Thus, SiB bootstraps secure communication, provides demonstrative identification of the device from which data originates, and captures user intentions in an intuitive way. What better way for a user to tell device A that it should communicate securely with device B than to take a picture of device B using device A's integrated camera?

In the remainder of this section, we detail the physical realization of the visual channel with 2D barcodes. Discussion of the visual channel's resilience against active attacks follows. The use of the visual channel to bootstrap secure communication is then illustrated with a specific example. We end this section with a discussion on using SiB with devices that may be lacking a display or a camera, or both. Sections 4, 5, and 6 then provide detailed usage scenarios for the demonstrative identification provided by SiB. In Section 7, we move on to discuss a weaker—though still valuable—property that can be provided by the visual channel, which we term *presence*.

3.1. 2D Barcodes as a Visual Channel

We implement the visual channel with a 2D barcode (e.g., Data Matrix [36], PDF417 [43], or MaxiCode [47]), displayed on or affixed to one device and captured by another with its digital camera. When a user executes the SiB protocol, she must aim the camera of her mobile device at a barcode on another device (either displayed electronically or affixed to the device's housing). The act of aiming the camera at the desired device results in demonstrative identification of the targeted device.

We now present a more detailed example of the use of SiB. Suppose Alice and Bob want to set up a secure channel between their camera phones. Alice's phone generates a 2D barcode encoding appropriate public cryptographic material and displays it on its screen, while Bob uses his phone's digital camera to take a snapshot of Alice's screen displaying the barcode. Bob must watch his phone's LCD, acting as viewfinder, updating in real time in response to his positioning of his camera-phone. A barcode recognition algorithm processes each image in the viewfinder in real time and overlays a colored rectangle around success-

fully recognized barcodes. When Bob presses the shutter button, the viewfinding process stops and the barcode recognition algorithm returns the data represented by the barcode. Section 8 presents further details of our implementation.

3.2. Attacking the Visual Channel

Active attacks are extremely difficult to perform against the visual channel without being detected by the user. The user has in mind the device at which she is aiming her camera, and will be conscious of a mistake if she takes a snapshot of anything else. We believe the act of taking a picture of *that* device—the one with which the user wants to communicate securely—is intuitive, and should therefore enjoy a low rate of operator error. Thus, the visual channel has the property of being resilient against active attacks (such as a man-in-the-middle attack), and the property that active attacks are easily detected by the user, who can then terminate wireless communication. It is ideal for authentication, providing the user with demonstrative identification of the communicating devices without burdening the user with device names or certificate management. We compare SiB to alternative means for authenticating devices in Section 9.2.

3.3. Pre-Authentication and the Visual Channel

We build on work by Balfanz et al. [4], and Stajano and Anderson [41], to secure wireless communication by leveraging the visual channel for authentication. We adopt the term pre-authentication, as Balfanz et al. suggest [4], to describe the data exchanged on the visual channel. Pre-authentication data is later used to authenticate one or both of the communicating parties in almost any standard public-key communication protocol over the wireless link. Eavesdropping on the visual channel gives no advantage to an attacker, provided that the underlying cryptographic primitives are secure, and that the mobile devices themselves have not been compromised.

Balfanz et al. discuss the use of infrared communication as a "secure side-channel" for pre-authentication between mobile devices [4]. They focus on the property that infrared is a "location-limited channel," emphasizing the difficulty an attacker faces in trying to interfere with the channel, because he must be in close physical proximity to the communicating devices. The primary advantage of SiB is that it uses a visual channel instead of an invisible channel, thus adding a direct human factor. We acknowledge that attacks against infrared are difficult to perform, but we believe that the inability of the user to actually see which devices are communicating provides dangerous opportunities to an attacker.

Figure 1 shows the pre-authentication phase of SiB, carried out over the visual channel. Provided that the mobile phone has not been compromised, and that the visual channel and relevant cryptographic primitives are secure against active adversaries (Section 9 presents a detailed security analysis), authentication in SiB requires merely that the user confirm her camera is pointed at the intended device.

3.4. Device Configurations

The concepts of SiB can be applied in different ways to devices with different capabilities, each equipped with either a camera and display, a camera only, a display only, or neither. In some cases, these device configurations impose some limitations on the strength of the achievable security properties. Figure 2 contains a summary of these properties.

The most flexible configuration for SiB is when both devices have both a camera and a display—these have a CD in their column or row heading in Figure 2. These devices can be mutually authenticated, since both possess cameras. Further, each device can make use of either a long-term public key or an ephemeral public key in each exchange, since barcodes containing keys are displayed on an electronic screen (as opposed to paper or some other fixed medium).

We refer to devices equipped with no display—devices which have no D in their column or row heading in Figure 2—as "displayless" devices. These devices can be authenticated with a long-term public key. A barcode encoding a commitment to the key, or multiple barcodes encoding the key itself, must be affixed to the device's housing (e.g., in the form of a sticker). The issue of whether to use a commitment to a key, or the key itself, is addressed in Section 5.

Entries in Figure 2 marked *presence* indicate that demonstrative identification of communicating devices is unattainable, but a property we term *presence* is still achievable. Presence refers to the ability to demonstrate that a device is in view of someone. We describe this property in more detail in Section 7.

4. Bidirectional Authentication

Providing mutual authentication between mobile devices that share no prior context is a difficult problem. In this section, we show how SiB can be used to intuitively capture user intentions and establish a mutually authenticated security context between precisely the devices the user wants, without a trusted authority. Examples of the established security context include authenticated ex-

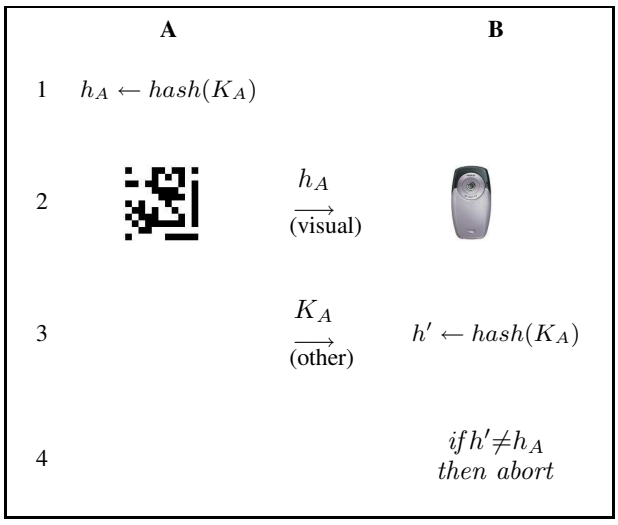

Figure 1. Pre-authentication over the visual channel. K_A is A's public key, which can be either long-term or ephemeral, depending on the protocol.

X \ Y	CD	C	D	N
CD	✓	✓*	✓	✓*
C	✓	✓*	✓	✓*
D	presence	presence	×	×
N	×	×	×	×

Legend	
✓	Strong authentication possible
✓*	Barcode label required on housing
presence	Confirm presence only
×	No authentication possible

Figure 2. Can a device of type X authenticate a device of type Y? We consider devices with cameras and displays (CD), cameras only (C), displays only (D), and neither (N).

change of public keys, and authenticated Diffie-Hellman key exchange to establish a shared secret. The device combinations we consider in this section are those where both devices have cameras.

We now walk through the use of SiB, beginning with device discovery and barcode generation. Next, we describe pre-authentication and bootstrapping a well-known public key protocol. Then, we describe options to satisfy different security requirements and project the likely performance of SiB on emerging mobile phones.

The SiB protocol begins when Alice and Bob decide they want to communicate securely. Alice's device first discovers nearby devices using its wireless interface. Alice then selects the device she believes to be Bob's (this is easily achieved with "friendly names" common to, e.g., Bluetooth [17]). Bob's device is listening when Alice makes a connection attempt.

Each user computes a commitment to their public key material and generates a barcode encoding this commitment. This key material can take the form of a user's long-term public key, or it can be an ephemeral key for use in only one key exchange. One practical example of this key material is a self-signed public key certificate extended with additional information about the key owner (e.g., name, email address, etc., similar to a vCard [10, 20]). The decision regarding what form of public key material to use is orthogonal to the authentication provided by SiB.

The pre-authentication phase now begins. The users take turns displaying and taking snapshots of their respective barcodes. The order is not important, but it is necessary that Alice capture the barcode commitment to Bob's public key, and that Bob capture the barcode commitment to Alice's public key. This pre-authentication protocol is secure as long as an attacker cannot find a second preimage for the commitment function, and is unable to perform an active attack on the visual channel.

After pre-authentication is complete, both devices now hold commitments to the other device's public key, and the devices can exchange public keys over the wireless link. The devices then perform the same commitment function over the other device's public key, ensuring that the result matches the commitment that was received over the visual channel. At this point, the devices have mutually authenticated one another's public keys, and Alice and Bob have demonstrative identification that the devices in their hands are the ones that are communicating. These authenticated public keys can then be used appropriately in any well-known public key protocol on the wireless link (e.g., IKE [19], SSL/TLS [11, 14]). It is imperative that the chosen established protocol verifies that each party does in fact hold the private key corresponding to their authenticated public keys.

The main contribution of SiB is the authentication provided by having demonstrative identification of the communicating devices. The selection of the well-known public key protocol is flexible. If a user desires to avoid transmitting a public key on the wireless network—so that eavesdroppers cannot ascertain which devices are communicating—the public key material can be encoded in a sequence of barcodes. The key is thereby obtained without transmitting it on the wireless medium, while re-

taining the demonstrative identification property with respect to the device originating the key. It is then advisable that the standard public key protocol that is used with SiB authentication is key-private [5].

As the processing and display capabilities of mobile phones improve, visual channel bandwidth will improve sufficiently for data transmitted over the visual channel to include network addresses for the relevant wireless interfaces (e.g., Bluetooth, 802.11) in addition to authentication data. This is more convenient for the user, since she never has to wait for discovery of neighboring devices or select a device from a list. Madhavapeddy et al. use barcodes on camera phones to improve the Bluetooth device discovery process [30].

5. Unidirectional Authentication

We now discuss entries from Figure 2 where the device of type X (the authenticator) is equipped with a camera, and the device of type Y (the device being authenticated) lacks a display and a camera. It is this presence of a camera on the authenticator, and lack of a display and a camera on the device being authenticated, that are responsible for the security properties of this particular device combination. We refer to a device of type X as camera-equipped, and a device of type Y as displayless.

Displayless devices do not have the ability to display newly generated values. Still, a camera-equipped device can authenticate displayless devices and establish secure communication channels. The displayless device must be equipped with a public/private keypair, and a sticker containing a barcode of a commitment to its public key must be affixed to its housing. Since the displayless device is constrained to the use of a single public/private key pair for its entire lifetime, the option to generate per-interaction public keys no longer applies. Of course, devices can be reprogrammed and new stickers affixed, but we consider that to be a significant maintenance task. As in Section 4, there are privacy issues with using fixed public keys that might be of concern.

We now introduce several practical examples where unidirectional authentication with SiB is useful. We first discuss connection to a wireless access point, and then describe securing the use of a printer in a public place. Finally, establishment of a trusted path for configuration of the Trusted Platform Module (TPM) in a TCG-compliant computing platform is considered.[2] A full discussion of the TCG specification is beyond the scope of this paper, though we summarize relevant concepts here as necessary.

[2]The TCG specifies a Trusted Platform Module (TPM) that can be used to enhance the security of computing platforms against software attacks [2, 3]. The TPM is a chip connected to a computer's processor, with no other I/O capabilities.

5.1. Practical Applications

An 802.11 access point (AP) is one example of a class of devices where "sticker-based" authentication may be desirable. Camera-enabled devices can authenticate the AP, enabling the establishment of a secure link-level connection from the camera-enabled device to the AP. This solution enables deployment of wireless connectivity in environments where security policies require physical presence for network access. Figure 3 shows the SiB application on a mobile phone scanning a barcode installed on a wireless access point.

Another commonly considered application where demonstrative identification of communicating devices is desirable is when using a printer in a public place. Similar to the wireless access point, the printer can have a barcode affixed to its housing so that a user can use SiB to authenticate wireless communication with the printer/print server and bootstrap establishment of a secure connection. Secure communication is important here not only to ensure the secrecy of the printed document, but to protect the user's computer from malicious software that masquerades as a printer driver.

Figure 3. Phone running SiB scanning a barcode on an 802.11 access point.

5.2. Establishing a Trusted Path for Configuration of a TPM

In this section we motivate the establishment of a trusted path to configure the TPM in a TCG-compliant

computing platform and then describe how SiB can be used to establish the trusted path. Today many computing platforms are plagued by "spyware" that may capture users' actions, including keystrokes, potentially exposing sensitive information like passwords and credit card numbers [39]. So far, we have assumed that users' devices are uncompromised. In this section, we relax this assumption with respect to the software running on a TCG-compliant computing platform, and discuss ways that SiB can aid in the establishment of a *trusted path* to configure a TPM.

One challenge in designing systems which incorporate a TPM is how a user can communicate securely with the TPM; the user has only a keyboard and display to communicate with the TPM, with untrusted operating system and window manager software in between. A TPM is configured—typically by a user or vendor—with a secret, the Owner Authorization Data (OAD), which can be used to exercise control over the TPM. A malicious party that captures the OAD (using, e.g., spyware) can change the OAD, delete (and in some instances change) application secrets in secure storage, and disable or enable TPM features at undesirable times. Unfortunately, in a TCG-compliant computing platform, if certain TPM features are disabled, a user has no way of knowing if malicious software is running and, e.g., logging all keystrokes. This is a serious problem if the user types the OAD while trying to configure the TPM—the malicious software has just captured the OAD.

Thus, it is undesirable to use the keyboard and display of a computing platform to configure the TPM for fear of malicious software running on the computing platform that can steal the OAD. In the remainder of this section, we show how authentication achievable with SiB enables the user to use her camera phone to send commands to the TPM, achieving secrecy so that a malicious application is unable to capture the OAD even if it has subverted the keyboard and display.

We propose the use of a camera phone to securely configure the TPM, where the user enters the OAD *only* on the camera phone. The TCG specifies that each TPM come equipped with a public / private *endorsement* keypair. The public endorsement key is used for encrypting commands to the TPM, and the private endorsement key is used exclusively for their decryption (it is never used to, e.g., compute a digital signature). Using SiB, the camera phone can authenticate a TPM's public endorsement key, and bootstrap secure communication with the TPM through which the user can enter the OAD and reconfigure the TPM as desired.

To enable TPM reconfiguration from a camera phone, a sticker must be installed on the case of the computing platform which contains either a barcode encoding a commitment to the public endorsement key, or several barcodes encoding the entire public endorsement key. In the case of a commitment, the full endorsement key can be obtained from the computing platform in an authenticated way analogous to the wireless access point example earlier in this section.

Note that an attacker with direct access to the computing platform can subvert the TPM by physical means. Thus, the use of SiB enhances security under the assumption of software-only attacks, which represent the majority of threats, and requires an attacker to have physical access to the computing platform, ruling out all remote attacks.

6. Screen Ownership and Application Integrity

In this section we consider the problem where a user wants assurance that what she sees on the screen of a TCG-compliant computing platform (e.g., a workstation or laptop) is really the content displayed by a particular application, and that this application is the same one used, e.g., last week—we want to determine that this application *owns* the screen. We propose to use SiB in conjunction with a TCG-compliant computing platform (hereafter: computing platform) to assist in determining whether the content displayed on a computing platform's screen truly originates from a particular application. In the context of TCG, SiB helps the user establish trust in the trusted computing platform in front of her. Our solution prevents a malicious application from impersonating a legitimate application on the computing platform. For example, the computing platform may have a virus that tries to spoof a legitimate control panel application to get the user to enter her password.

The TCG specifies that the TPM has Platform Configuration Registers (PCRs) that a properly instrumented operating system *extends* with hash values computed over all software that is loaded for execution [45]. The purpose of the PCRs is to enable the TPM to digitally sign attestations that a particular software configuration has been loaded. The OS must also store *history information* that provides additional information about a computing platform's configuration (code or relevant computing platform-specific information that has been reported to a computing platform's TPM) to a requesting entity referred to as a *challenger*. This *history information* is managed by the Trusted Platform Support Service (TSS), generally a piece of the OS, that need not be trusted because the *history information* can be validated using the PCR values. We leverage these properties of a TCG-compliant computing platform to demonstrate screen ownership with SiB.

We include a brief note on terminology. The TPM is

a passive component, so application-level commands do not actually get sent to the TPM—they are interpreted by a piece of trusted software that can be attested to, usually a module in the operating system, which is distinct from the TSS. This trusted software translates the application-level commands into TPM hardware instructions. Subsequently, when we say the user sends a command "to the TPM," we mean that it is sent to this special OS module, and, if necessary, the TPM can be used in an attestation that the OS module has not been modified. A TCG-compliant computing platform requires OS support as well as a physical TPM, and we refer the interested reader to [3] for more information.

6.1. Windowing System Requirements

Several prerequisites must be met to verify that the desired application actually owns the screen. The operating system and the windowing system (i.e., the window manager) must be designed such that they can enforce the condition that the desired application's window can be put into an "always-on-top" mode. The application must be notified by the windowing system that it is in this mode, and the application must be subsequently notified if the windowing system takes the application out of the "always-on-top" mode before being requested to do so by the application. In today's popular windowing systems, notification events of this nature may be dropped if the system is under heavy load. It is imperative for the security of this verification protocol that these notification events are guaranteed to arrive.

The window manager must be able to prevent other applications from reading the contents of the screen where the user's desired application is "always-on-top." This includes such cases as taking screenshots and copy-and-paste operations. All pixels displayed by the user's desired application when it is in "always-on-top" mode must be inaccessible by any other application. Shapiro et al. have taken a step towards achieving the necessary properties with the EROS trusted window system [40].

We achieve through demonstration of screen ownership the property that the integrity of a particular application can be verified. For example, we can verify that the current application is, e.g., the same application we were using last week. This is different from the property of confirming that the current application is, e.g., precisely the application purchased from a particular software vendor. The techniques described below can easily be extended to achieve such a property, however, additional vendor-specific configuration is necessary on the user's camera phone.

In this approach, the user's application need not be certified. The spirit of operation is similar to that of a common SSH session—upon first connection keys are exchanged under the assumption that an attack is highly unlikely. If this assumption holds, then all subsequent connections are secure.

6.2. Initial Configuration

We now describe the basic operation of SiB on a TCG-compliant computing platform in two parts—the initial configuration, and subsequent verifications. We omit many details for brevity (again, we refer the interested reader to the relevant chapters of [3]).

We explain the initial configuration under the assumption that there are no software attacks in progress on the computing platform being configured. SiB does not require the wireless network to be secure. We assume the user has already established a secure channel to the TPM via Bluetooth, as Section 5.2 explains. The user then elects to measure an initial configuration of the on-top application on the computing platform by selecting the appropriate menu item on her mobile phone. Upon reception of the command to measure an initial configuration of the active application in the TPM-controlling piece of the OS, the following occurs:

1. The window manager temporarily locks out other applications, and ensures that the active application remains "always-on-top" for the duration of the protocol. The "always-on-top" mode must meet the requirements detailed in Section 6.1. This is to ensure that the user is not confused concerning the application with which she is interacting.

2. The TPM allocates a new secure key object and generates a new public / private signing key-pair $(K_{verify}, K_{verify}^{-1})$ to be used in subsequent verifications of the active application. This key is "wrapped" by the TPM and bound to the current values of the PCR registers for the OS, window manager, and active application [45]. The appropriate *history information* from the TSS must also be collected.

3. The application being configured for future verification uses SiB to generate and display a barcode commitment to the public K_{verify}. The public key is sent to the user's mobile phone, along with appropriate *validation* data (digitally signed with the new K_{verify}^{-1}) constructed from the current values of the PCR registers and the relevant *history information*. Meta-data should also be sent to the user's mobile phone such that the user can easily select the correct application for performing subsequent verifications. The user authenticates the public key using the

barcode displayed onscreen with SiB, i.e., by photographing the barcode with her camera phone as in Section 5.

6.3. Subsequent Verification

We leverage the ability of the TPM to conditionally allow the use of a wrapped key [3]. The condition is that the appropriate PCR registers contain the same values that they did upon initial configuration. That is, the same operating system, window manager, and application are running now that were running during initial configuration. The details of this key release procedure involve TCG-specific OS capabilities for handling the *history information* in response to *challenges*, which we omit here.

To verify an application, the user selects verification of the appropriate entry from a list of applications on her mobile phone (an entry is added to this list during each initial configuration, we omit the details for brevity). The verification then proceeds as follows:

1. The phone generates a cryptographically secure nonce, $nonce$, and sends it via Bluetooth to the desired application running on the computing platform. This nonce serves as a cryptographic challenge to the computing platform; it need not be kept secret.

2. We assume the nonce reaches some application. To successfully complete this protocol, that application must then sign the nonce using K^{-1}_{verify}. Since the TCG-compliant OS knows the state of the computing platform, only the correct application will meet the requirements for the TPM to perform the signature (the appropriate PCR registers for the specified application, window manager, and OS contain the values recorded during the initial configuration).

3. If the correct values are present, the TCG-compliant OS will then grant the application the privilege to use the private verification key K^{-1}_{verify}.

4. The TPM will then sign the challenge using the private verification key on behalf of the application: $\sigma \leftarrow Sign_{K^{-1}_{verify}}(nonce)$.

5. The application encodes σ in one or more barcodes and displays those on the screen. Since the computing platform's windowing system meets the requirements in Section 6.1, a malicious application running on the computing platform will never be able to read these barcodes.

6. The user uses the SiB component of the application on her camera phone to capture this signature and verify it with K_{verify} as obtained in the initial configuration: $Verify_{K_{verify}}(nonce, \sigma)$.

7. If the signature verifies, the user has assurance that the application displayed on the screen of her computing platform is precisely the same application that was active when she performed the initial configuration.

Note that malicious software executing on the computing platform being verified may be able to capture the challenge, but it cannot access the TPM-protected private key K^{-1}_{verify} necessary to compute a valid signature. Upon execution of the malicious software, the PCR values on the computing platform change, eliminating the ability to access K^{-1}_{verify}. It is even possible to protect against vulnerabilities (e.g., buffer overflows) in the application itself. Sailer et al. detail an integrity measurement architecture that provides such properties [38].

If either of the user's mobile phone or computing platform is not equipped with a Bluetooth interface, we can still achieve strong security properties at the expense of convenience for the user. The role of Bluetooth in the protocol is to deliver the cryptographic challenge to the TPM. Without Bluetooth, the user must type the challenge—or input it in some other way, such as with a camera attached to the computer—displayed on her phone into the application on her computer that currently owns the screen. The verification protocol then proceeds as before from step 2.

7. Presence Confirmation

A display-only device (display-equipped and cameraless) is unable to strongly authenticate other devices using SiB. Equipped with no camera, it makes no difference whether the entity the cameraless device wants to authenticate has a display, or makes use of a barcode sticker—the cameraless device cannot "see" them. However, display-only devices can obtain a property we refer to as *presence* (see Figure 2). That is, it can confirm the presence of some other device in line-of-sight with its display.

To detect the presence of a nearby device, the display-only device generates a key K for a message authentication code (MAC), encodes it in a barcode, and displays that barcode, noting the time when it was first displayed. Any nearby devices that are able to see the display and capture the barcode can send data to the display along with a MAC computed over that data, $\{data, MAC(K, data)\} \rightarrow display - only\ device$.

When the data and MAC arrive over the wireless channel, the display-only device knows that some device has been in line-of-sight during the time since K was first displayed. We emphasize that this *presence* property is quite

weak—the display-only device has no way of knowing how many devices can see its display, or whether the radio signal is from the same device that is in line-of-sight with its display. It can only verify the MAC it receives with the data over the wireless channel, and it can measure the delay between displaying the barcode and receiving the MAC on the wireless channel.

Despite the weakness of the presence property, there are still practical applications for devices capable of determining presence. For instance, the presence property is useful in the context of a smart home. It can restrict remote control access on a television to users in the same room. In general, it can serve to limit authority to control a device to users located in view of that device.

Consider the establishment of a security context between a TV and a DVD player to secure wireless communication between the two. The user can use SiB to strongly authenticate the DVD player to her phone through a barcode attached to the DVD player's housing. She can then demonstrate the DVD player's presence to the TV by sending it the public key of the DVD player, along with a MAC over the DVD player's public key, $\{K_{DVD}, MAC(K, K_{DVD})\} \rightarrow TV$.

The TV is then configured to establish a secure, authenticated connection to the DVD player whenever the user selects DVD player as the input source on the TV. Taken one step further, the TV can add the DVD player to its list of trusted devices, such that the TV will automatically accept input from the DVD player whenever the user inserts a DVD.

8. Implementation Details

8.1. Series 60 Phone Application

We built SiB in C++ such that it will run on mobile phones running Symbian OS versions 6.1 and 7.0s with the Nokia Series 60 User Interface. The size of the Symbian Installation System (SIS) file [44] for SiB is only 35 kilobytes. This makes deployment feasible over even the most constrained channels, such as GPRS [37].

The Nokia 6600 and 6620 served as our development platforms. The barcode format and image processing algorithm in our system is adapted from that of Rohs and Gfeller [34]. The data contained in the barcodes for SiB is augmented with Reed-Solomon error correcting codes to provide better performance in the presence of errors in the image processing [33]. We ported Karn's implementation of Reed-Solomon codes to Symbian OS [22].

In its current state, we use the SHA-1 cryptographic hash function for all hashing operations [21]. All wireless communication occurs via Bluetooth [17]. To enable users to perform a key exchange between two camera phones, we implemented ephemeral Diffie-Hellman key exchange to establish a shared secret between the two devices [12], as discussed in Section 4. This shared secret can be used to establish an authenticated channel over the wireless data link between the two devices, over which any desired information can be exchanged.

Since the Reed-Solomon codes embedded in the barcode indicate whether a processed code is valid or invalid, and our application constantly decodes any barcodes in the current camera scene, it is not strictly necessary for the user to press a "shutter" button for the camera. Our implementation is configurable so that the user may elect whether she wishes to press a shutter button. When the shutter button is disabled, the first valid image processed can be used automatically. However, if the user is in an environment where there are many barcodes, recognition of the incorrect barcode will cause the SiB protocol to abort. For particularly cautious users, who may be concerned that automatic shutter control could cause the camera to capture a barcode displayed by a malicious party on another medium before the user aims the camera at the desired barcode, manual control of the shutter is possible.

Figure 4 contains a photograph of SiB in action. Alice's Nokia 6620 (background), is displaying a barcode, while Bob's Nokia 6620 (foreground) is successfully decoding the data encoded in Alice's phone's barcode. In bidirectional authentication with SiB, Alice and Bob would then switch roles. Bob's phone would display a barcode, and Alice's phone would decode it. Figure 5 contains screen shots of the SiB application. In Figure 5(a), the user is being prompted to select the desired Bluetooth device for the key exchange. We emphasize that device selection is for convenient establishment of the wireless channel only, and is not a factor in the security of SiB. Madhavapeddy et al. detail the encoding of Bluetooth network addresses as barcodes displayed on mobile phones to eliminate the Bluetooth discovery process [30]. Implementation of SiB using a barcode with sufficient bandwidth to encode a Bluetooth network address and a cryptographically secure hash value can eliminate the manual device selection process. Figure 5(b) shows the display of one phone when it has successfully recognized the barcode displayed on the screen of another phone.

8.2. Barcode Reading Performance

In this section we evaluate the performance of our SiB application on the Nokia 6600. We consider the length of time required for SiB to mutually authenticate two mobile phones and perform a Diffie-Hellman key exchange, as in Section 4. We then provide insight into the practicality of using multiple barcodes to encode a single logical item.

Figure 4. SiB application on a Nokia 6620 with one phone scanning the hash-barcode on the LCD of another.

(a) Device selection.

(b) Barcode recognition.

Figure 5. Mobile phone screen shots showing the SiB application in operation. In the first screen shot, the user is being prompted to select a device with which to initiate the SiB protocol. In the second, the user sees the phone at which she is aiming her camera, with status markers indicating successful barcode recognition.

For example, a 1024 bit RSA key would need to be encoded in 16 barcodes in our current implementation—64 bits of key and 4 bits of place-marker in each barcode.

We performed timing analysis on our implementation of bidirectional authenticated Diffie-Hellman key exchange application (see Section 4) between two Nokia 6600s. We instrumented the application to track the length of time between the establishment of the Bluetooth connection and the successful completion of the Diffie Hellman key exchange. It is reasonable to ignore Bluetooth device discovery since a commercially viable implementation should use a barcode format with sufficient bandwidth (e.g., [36, 49]) to include the Bluetooth address of the displaying device, rendering device discovery unnecessary. Over a series of 10 executions involving manual aiming of the phones, we observed the average length of time to establish a shared secret to be 8 seconds. The minimum and maximum times ranged between 6 and 10 seconds, respectively. The processing time required by SiB is roughly one second. The execution time of SiB is dominated by the users aiming their cameras at the correct barcodes.

On the Nokia 6600, SiB is able to process two to three barcode snapshots per second. We have successfully read in excess of five barcodes from a single snapshot, for a sustainable rate averaging 10 to 15 barcodes per second under ordinary office lighting conditions (see Figure 6). Thus, we conclude that reading multiple barcodes for a single logical item is a viable implementation strategy.

9. Security Analysis

In addition to the security of the underlying cryptographic primitives, the security of SiB is based on the assumption that an attacker is unable to perform an active attack on the visual channel, and is unable to compromise the mobile device itself. We first discuss the employed cryptographic primitives, then the security properties of various side-channels for authentication. Finally, we discuss the security properties of authentication using a visual channel.

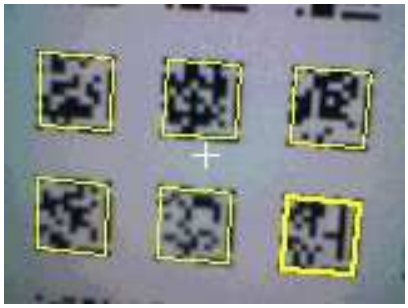

Figure 6. Mobile phone screen shot showing the SiB application recognizing multiple barcodes displayed on an LCD screen.

9.1. Cryptography

As implemented, each barcode in SiB has a raw bandwidth of 83 bits. 15 of these bits are dedicated to Reed-Solomon error correcting codes, leaving the application with 68 bits of hash for authentication. As discussed in Section 4, the hash transmitted in the barcode needs to be secure against active attacks, which we achieve through the properties of the visual channel. However, if an adversary can find a second pre-image of the value encoded in the barcode, then a passive attack on the barcode coupled with an active attack against the wireless network connection can be successful. As mobile phone cameras and displays increase in fidelity, the key itself can be encoded in the barcode, eliminating this dependence on a cryptographic hash function.

For a cryptographic system to be considered computationally secure today, at least 2^{80} operations should be required to break it [26]. When the barcode containing a key's digest takes the form of a sticker on the side of some device, or gets displayed on a laptop or other relatively large screen, it is not a problem to use multiple barcodes to encode sufficient cryptographic material to be secure. As detailed in Section 8.2, SiB is capable of reading multiple barcodes simultaneously.

However, in phone-to-phone key establishment, there is only enough screen area to display a single barcode. Thus, the 68 bits of hash data in our key exchange prototype are potentially vulnerable to cryptanalytic or brute-force attack. However, this vulnerability is significant only when the barcode represents a hash of a long-term public key. We propose two methods for circumventing this problem. The first simply involves the use of multiple barcodes to encode sufficiently many bits of hash, displayed in sequence. The phones can synchronize the displaying and reading via Bluetooth messages so that the user is not inconvenienced by pressing buttons to iterate through multiple barcodes. While effective, this technique adds to the burden the user must face.

We propose a second technique which still achieves strong security properties through the use of Diffie-Hellman session keys. The devices participate in an ephemeral Diffie-Hellman key exchange, using SiB to authenticate their respective public contributions to the key. This session key is then used to establish an authenticated channel over which the devices may exchange all manner of data. Once the DH session-key establishment is complete, there is no advantage to finding a pre-image or second pre-image of the hash function.

Even with today's technology, computing 2^{67} operations takes well over a few seconds, which is the length of time DH key establishment using SiB for authentication requires to execute. Once a session key is established, the attacker gains nothing even if he compromises the hash. In a commercially viable system, the barcode generator and recognition algorithm should be extended to achieve a useful data content of 80 bits or more. There are no major technical obstacles preventing this extension.

9.2. Selecting an Authentication Channel

Mutual authentication between two parties without the assistance of a trusted authority requires a channel that is secure against active attacks, such as a man-in-the-middle (MITM) attack. We analyze potential channels based on the degree to which the user's intentions are captured, and the amount of feedback that the channel provides to the user. Figure 7 contains a summary of proposed channels and their characteristics.

Activity on channels such as infrared, ultrasound, or radio is undetectable to humans without specialized equipment. Therefore, if Alice believes her device is communicating with Bob's device via infrared, the only assurance she has that it is actually doing so is through status indicators on the two devices. She cannot see infrared radiation leaving her device and entering Bob's, and she certainly cannot see an attacker's device outputting interference patterns and affecting the data stream. Similarly, in case of ultrasound and radio, Alice and Bob need to rely on status indicators of their devices, but they are not sure that Alice's device is indeed setting up a key with Bob's device. Thus, the users' intentions are not captured well, and feedback is indirect and prone to error. Using an audible signal (marked "beeps" in Table 7) for data exchange is more intuitive, but this would not work well in noisy environments and is still prone to a man-in-the-middle attack since it can be difficult for people to tell where "beeps" originate and how many devices are "beeping."

Physical contact between devices is much more intuitive for people and captures the intentions of the users—

Channel	COTS	Resists MITM	Convenient
Ultrasound	○	○	●
Audible ("beeps")	○	◐	●
Radio	●	○	●
Physical Contact	○	●	●
Wired Link	●	●	○
Spoken Passwords	N/A	●	○
Written Passwords	N/A	●	○
Visual Hash Verif.	●	●	◐
Infrared	●	◐	◐
Seeing-Is-Believing	●	●	●

Figure 7. Characteristics of various channels proposed for authentication. We acknowledge that rating the convenience of a channel is subjective; however, we believe it is useful to compare various channels in this way. COTS indicates that the necessary hardware is already present in Commercial Off-The-Shelf products. Symbols: yes (●), partial (◐), no (○).

identifying the devices between which they want to establish a secure communication link [41]. Unfortunately, most current devices are not equipped with an interface for this purpose. An alternative approach is to use a wired link, for example connect both devices with a USB cable, however, this approach is not convenient to use and people would need to carry a wire with them.

Another approach is for Alice and Bob to establish a secret password, either by speaking the password aloud, or by writing passwords on paper and passing them to each other. Both Alice and Bob would then need to type in the password correctly, which the devices use to perform a secure password protocol, e.g., EKE [6]. We believe this approach is cumbersome in comparison with SiB, particularly on devices with a limited keyboard.

Finally, both devices could present a visual representation of the hash of the established Diffie-Hellman key to detect a man-in-the-middle attack [13, 16, 27, 32]. These approaches, however, are not secure unless people carefully compare the output of the visual hash function. We believe SiB has an advantage here not just in ease-of-use but because strong authentication is intrinsically linked with device identification.

9.3. Attacks Against Seeing-Is-Believing

In Section 6 we showed how to achieve screen ownership and application integrity on a TCG-compliant platform. One weakness of our approach is that it provides an instantaneous guarantee only. Once the application being verified (the user's desired application) is no longer in "always-on-top" mode, a malicious application may be able to tamper with the input or output of the desired application. This problem has been studied in the context of secure windowing systems, e.g., [40], and is dependent on window system policy. One solution is to leave the application in always-on-top mode for the duration of its use. In the context of TCG, if the user's task is sufficiently sensitive, she may only be willing to perform it when the platform is in a "dedicated trusted state," as detailed in [3].

Attackers may attempt to construct a window that obscures all of the desired application except its barcode, confusing the user into believing the malicious application allowing the legitimate barcode to show through is actually the desired application. This attack is prevented by using SiB with an application on the platform only when that application is in "always-on-top" mode.

Attackers may also attempt to read the pixels of the user's desired application, identify which pixels represent the barcode, and redisplay the legitimate barcode in a malicious application that appears legitimate to the user. To prevent this attack, the window manager must be able to prevent other applications from reading the contents of the screen where the user's desired application is "always-on-top." This includes such cases as taking screenshots and copy-and-paste operations. All pixels displayed by the user's desired application when it is in "always-on-top" mode must be inaccessible by any other application. Shapiro et al. have taken a step towards achieving the necessary properties with the EROS trusted window system [40].

In Section 7, we discuss a presence property which requires the user to demonstrate that it can see a display. Kuhn details some attacks which enable a malicious party to read the contents of a CRT screen without actually being in line-of-sight with it. For example, a sophisticated adversary may be able to measure the electromagnetic radiation emitted by the CRT [25]. Alternatively, an adversary may be able to assemble the contents of the CRT by looking at reflected light from the CRT [24]. Defense against this form of attack is outside the scope of SiB.

An attacker can disrupt the lighting conditions around Alice and Bob in an attempt to disrupt SiB. However, changes of sufficient magnitude to impair SiB are easily observed by Alice, Bob, and any people in the vicinity, alerting them to some kind of unusual behavior. A more sophisticated, and subtle, attack is to use infrared radiation to overwhelm the CCD[3] in a phone's camera. If an

[3]Charge Coupled Devices (CCDs) are the prevalent type of image sensor used in today's digital cameras.

attacker is able to flood an environment with sufficient infrared radiation, the CCD in a phone's camera can begin to saturate, and all attempts to take pictures will yield a picture with all pixels set at or above the intensity of the legitimate image, up to the maximum value for each pixel. Essentially, the image becomes noise. Alice will see that the image in her viewfinder is not the picture of Bob's phone that she expects, and can abort the protocol.

Even without a user monitoring the process, the electronic-warfare-esque techniques necessary to cause the CCD to output something other than the scene in front of the camera are beyond the reach of all but the most sophisticated adversaries with current technology. We are unaware of any attacks feasible today which result in anything but noise from the camera under attack.

10. Conclusion

In this paper we propose Seeing-Is-Believing, a system that uses barcodes and camera phones as a visual channel for human-verifiable authentication. This channel rules out man-in-the-middle attacks against public-key based key establishment protocols. The visual channel has the desirable property that it provides demonstrative identification of the communicating parties, providing the user assurance that her device is communicating with *that* other device. SiB enables establishment of a trusted path for configuration of the TPM in a TCG-compliant computing platform. Leveraging a TCG-compliant computing platform and SiB, one can verify the integrity of an application over multiple invocations. We have also analyzed the establishment of secure, authenticated sessions between SiB-enabled devices and devices missing either a camera, a display, or both, and found that secure communication is possible in many situations.

11. Acknowledgements

We are indebted to the following people for their insightful comments and helpful discussions: Michael Abd-El-Malek, Doug Baker, Lujo Bauer, Leendert van Dorn, Simson Garfinkel, Jason Rouse, and Jesse Walker. We would also like to thank the anonymous reviewers for their valuable suggestions.

References

[1] CoolTown. `http://www.cooltown.com/`, Nov. 2004.

[2] Trusted Computing Group. `http://www.trustedcomputinggroup.org/`, Nov. 2004.

[3] B. Balacheff, L. Chen, S. Pearson, D. Plaquin, and G. Proudler. *Trusted Computing Platforms – TCPA Technology in Context*. Prentice Hall, 2003.

[4] D. Balfanz, D. Smetters, P. Stewart, and H. C. Wong. Talking to strangers: Authentication in ad-hoc wireless networks. In *Proceedings of the Symposium on Network and Distributed Systems Security (NDSS)*, Feb. 2002.

[5] M. Bellare, A. Boldyreva, A. Desai, and D. Pointcheval. Key-privacy in public-key encryption. In *Proceedings of Advances in Cryptology (ASIACRYPT)*, 2001.

[6] S. Bellovin and M. Merrit. Augmented encrypted key exchange: a password-based protocol secure against dictionary atttacks and password file compromise. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, pages 244–250, 1993.

[7] S. M. Bellovin and M. Merrit. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 72–84, 1992.

[8] V. Boyko, P. MacKenzie, and S. Patel. Provably secure password authentication and key exchange using Diffie-Hellman. In *Proceedings of Advances in Cryptology (EUROCRYPT)*, pages 156–171, 2000.

[9] M. Burnside, D. Clarke, B. Gassend, T. Kotwal, M. Burnside, S. Devadas, and R. Rivest. The untrusted computer problem and camera-based authentication. In *Proceedings of Pervasive Computing (Pervasive)*, Aug. 2002.

[10] F. Dawson and T. Howes. RFC2426: vCard MIME directory profile. `http://www.faqs.org/rfcs/rfc2426.html`, Sept. 1998.

[11] T. Dierks and C. Allen. RFC 2246: The TLS protocol: Version 1.0. `http://www.faqs.org/rfcs/rfc2246.html`, Jan. 1999.

[12] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Trans. Inform. Theory*, IT-22:644–654, Nov. 1976.

[13] S. Dohrmann and C. Ellison. Public key support for collaborative groups. In *Proceedings of the PKI Research Workshop*, Apr. 2002.

[14] A. O. Freier, P. Karlton, and P. C. Kocher. The SSL protocol: Version 3.0. `http://wp.netscape.com/eng/ssl3/draft302.txt`, Nov. 1996.

[15] E. Gieseke and J. McLaughlin. Secure web authentication with mobile phones using keyed hash authentication. CSCI E 170 Final Project, Harvard University Extension, Jan. 2005.

[16] I. Goldberg. Visual key fingerprint code. `http://www.cs.berkeley.edu/iang/visprint.c`, 1996.

[17] J. C. Haartsen. The Bluetooth radio system. *IEEE Personal Communications Magazine*, pages 28–36, 2000.

[18] S. R. Hanna. Configuring security parameters in small devices. `draft-hanna-zeroconf-seccfg-00.txt`, July 2002.

[19] D. Harkins and D. Carrel. RFC2409: The Internet key exchange (IKE). `http://www.faqs.org/rfcs/rfc2409.html`, Nov. 1998.

[20] T. Howes and M. Smith. RFC2426: MIME content-type for directory information. `http://www.faqs.org/rfcs/rfc2425.html`, Sept. 1998.

[21] P. Jones. RFC3174: US secure hash algorithm 1 (SHA-1). http://www.faqs.org/rfcs/rfc3174.html, Sept. 2001.

[22] P. Karn. Reed-solomon encoding/decoding. http://www.ka9q.net/code/fec/, 2002.

[23] L. M. Kohnfelder. Towards a practical public-key cryptosystem. B.Sc thesis, MIT Departement of Electrical Engineering, 1978.

[24] M. G. Kuhn. Optical time-domain eavesdropping risks of crt displays. In *Proceedings of the IEEE Symposium on Security and Privacy*, May 2002.

[25] M. G. Kuhn and R. J. Anderson. Soft tempest: Hidden data transmission using electromagnetic emanations. In *Proceedings of the Information Hiding Workshop (IHW)*, pages 124–142, Apr. 1998.

[26] A. K. Lenstra and E. R. Verheul. Selecting cryptographic key sizes. *Journal of Cryptology*, 14(4):255–293, 2001.

[27] R. Levien. PGP snowflake. Personal communication, 1996.

[28] P. MacKenzie, S. Patel, and R. Swaminathan. Password authenticated key exchange based on RSA. In *Proceedings of Advances in Cryptology (ASIACRYPT)*, pages 599–613, 2000.

[29] A. Madhavapeddy, D. Scott, R. Sharp, and E. Upton. Using camera-phones to enhance human-computer interaction. In *Proceedings of Ubiquitous Computing (Adjunct Proceedings: Demos)*, 2004.

[30] A. Madhavapeddy, D. Scott, R. Sharp, and E. Upton. Using visual tags to bypass bluetooth device discovery. In *Proceedings of the ACM Mobile Computing and Communications Review (MC2R)*, Jan. 2005.

[31] S. P. Miller, C. Neuman, J. I. Schiller, and J. H. Saltzer. Kerberos authentication and authorization system. In *Project Athena Technical Plan*, page section E.2.1, 1987.

[32] A. Perrig and D. Song. Hash visualization: A new technique to improve real-world security. In *Proceedings of the Workshop on Cryptographic Techniques and E-Commerce (CrypTEC)*, pages 131–138, July 1999.

[33] I. S. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics*, 1960.

[34] M. Rohs and B. Gfeller. Using camera-equipped mobile phones for interacting with real-world objects. *Proceedings of Advances in Pervasive Computing*, pages 265–271, Apr. 2004.

[35] S. Ross, J. Hill, M. Chen, A. D. Joseph, D. Culler, and E. Brewer. A composable framework for secure multimodal access to Internet services from post-PC devices. In *Mobile Networks and Applications (Special Issue: Selected papers from WMCSA 2000)*, pages 389–406, Oct. 2002.

[36] RVSI Acuity CiMatrix. Data matrix barcodes. http://www.rvsi.com/acuitycimatrix/index.htm, 2005.

[37] P. Rysavy. General packet radio service (GPRS). *GSM Data Today*, Sept. 1998.

[38] R. Sailer, X. Zhang, T. Jaeger, and L. van Doorn. Design and implementation of a TCG-based integrity measurement architecture. In *Proceedings of the USENIX Security Symposium*, 2004.

[39] S. Saroiu, S. D. Gribble, and H. M. Levy. Measurement and analysis of spyware in a university environment. In *Proceedings of the Symposium on Networked Systems Design and Implementation (NSDI)*, Mar. 2004.

[40] J. S. Shapiro, J. Vanderburgh, E. Northup, and D. Chizmadia. Design of the EROS trusted window system. In *Proceedings of the USENIX Security Symposium*, pages 165–178, 2004.

[41] F. Stajano and R. Anderson. The resurrecting duckling: Security issues for ad-hoc wireless networks. In *Proceedings of the Security Protocols Workshop*, 1999.

[42] J. G. Steiner, C. Neuman, and J. I. Schiller. Kerberos: An authentication service for open network systems. Manuscript, Mar. 1988.

[43] Symbol Technologies. PDF417. http://www.pdf417.com/, Feb. 2005.

[44] A. Thoukydides. SIS file format. http://homepage.ntlworld.com/thouky/software/psifs/sis.html, 2004.

[45] Trusted Computing Group. Trusted platform module main specification, Part 1: Design principles, Part 2: TPM structures, Part 3: Commands. http://www.trustedcomputinggroup.org, Oct. 2003. Version 1.2, Revision 62.

[46] J. D. Tygar and A. Whitten. WWW electronic commerce and Java Trojan horses. In *Proceedings of the USENIX Workshop on Electronic Commerce*, pages 243–250, Nov. 1996.

[47] United Parcel Service. MaxiCode. http://www.maxicode.com/, Feb. 2005.

[48] S. Čapkun, J. Hubaux, and L. Buttyán. Mobility helps security in ad hoc networks. In *Proceedings of the ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, June 2003.

[49] S. Woodside. Read real-world hyperlinks with a camera phone. http://semacode.org/, Feb. 2005.

[50] M. Wu, S. Garfinkel, and R. Miller. Secure web authentication with mobile phones. http://dimacs.rutgers.edu/Workshops/Tools/abstract-wu-garfinkel-miller.%pdf.

[51] T. Wu. The secure remote password protocol. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, Feb. 1999.

[52] P. Zimmermann. *The Official PGP User's Guide*. MIT Press, 1995.