Jeu de Paume: A Security Suite for Handheld Devices

Niklas Frykholm[1]* and Markus Jakobsson[2] and Ari Juels[2]

[1] niklas@kagi.com
[2] RSA Laboratories, 174 Middlesex Turnpike, Bedford, MA 01730,
{mjakobsson,ajuels}@rsasecurity.com

Abstract. We describe *jeu de paume*, a small suite of security tools specially geared toward the protection of data on handheld devices. The components of jeu de paume are: (1) A visual password system adapted to stylus-based data entry; (2) An encryption system for credit-card numbers using low-entropy keys; and (3) A signature scheme with a cryptographic mechanism for deterring theft.

Key words: digital signing, encryption, graphical passwords, handheld-device security, palm devices, visual passwords

1 Introduction

Decreasing size and increasing power are rapidly extending the pervasiveness of handheld consumer computing devices. Palm devices, as we refer to them in this paper, are now capable of storing large amounts of information and performing functions requiring a good deal of computing power, such as generation of digital signatures. Indeed, the most recent generation of palm devices offers much of the functionality associated only a few years ago with desktop computers, including powerful operating systems and color screens. With the benefits of portability and power, however, comes a drawback: Palm devices are very easily lost or stolen. This is especially problematic given the range and sensitivity of the data that users increasingly rely on palm devices to store, including appointment and address books, credit card numbers, and even passwords.

In this paper, we describe a suite of information security tools that we refer to collectively as *jeu de paume*. ("Jeu de paume" literally means "palm game". The name refers to an adversarial sport played in France several centuries ago, and recognized as an ancestor of modern-day tennis. In our work, we maintain the adversarial aspect of the game, but strive to guarantee loss for the adversary.) The aim of jeu de paume is to protect the data in lost or stolen palm devices against compromise.

* This work performed while the author was at RSA Laboratories, Stockholm, Sweden.

Palm security: Old concerns in a new setting: The questions we address (such as password-secured login and secure encryption) have received extensive treatment in the literature, with a wide range of proposed solutions. Most of the proposed security mechanisms, however, have focused on traditional, stationary computing systems and do not work well for palm devices, which are subject to a somewhat different spectrum of threats.

One issue of importance is that the design of palm devices specifically makes synchronization easy, an advantage that has to be considered from the perspective of an attacker as well as that of the honest user. An adversary that successfully manages to synchronize with a stolen or otherwise compromised device can transfer its complete state to a hostile environment. This special form of vulnerability is analogous to but more comprehensive than compromise of larger computing devices by Trojan horse or viruses.

Another important consideration is the particular nature of data entry on palm devices: Use of a stylus or numerical keypad makes text entry more cumbersome than with than a keyboard. This turns into a security problem if users decide to circumvent or cut corners with security mechanisms due to their inconvenience. The problem is aggravated by the fact that palm devices typically are used in "short bursts" (unlike standard computers), which reduces user tolerance to hassles and delays.

As a result, users tend to employ passwords and PINs on palm devices that are even weaker and more poorly managed than in traditional setttings, i.e., of lower effective entropy. In stationary environments, the use of domain passwords and other server-assisted security mechanisms helps compensate to some extent for weak passwords. On palm devices, no such aids may be assumed to be consistently available. This renders the design of good security architectures for lost, temporarily misplaced, or stolen palm devices particularly challenging.

Our security tools: Let us now give a summary description of the security tools in jeu de paume:

1. **Visual passwords:** Our password-entry system is specially geared to the small screens and stylus-based data entry common in palm devices. A visual password (or visual PIN) in jeu de paume consists of a set of distinguished points on multiple images. Visual cues, capacity for error-tolerance, and an initial training routine enable users to retain visual passwords better than text-based ones, we believe. The fact that the system is stylus based also allows users to enter visual passwords more easily than text-based ones on palm devices.
2. **Credit-card vault:** The credit-card vault in jeu de paume is a file encryption system designed specifically for use with credit card or bank account numbers. By exploiting a form of non-redundant encryption, this system provides good security even with a very low-entropy encryption key.
3. **Funkspiel signatures:** Funkspiel signatures similarly represent a way of protecting a digital signing key with a low-entropy PIN. The basic notion is to have the user provide a PIN at the time of signing. This PIN is embedded

in the digital signature in such a way that it can be checked by some trusted third party, such as a bank. At the same time, an adversary with access to many signing transcripts (even in an active attack) cannot feasibly determine the PIN, or verify the correctness of a guess.

We note that the credit card vault and the funkspiel signatures both can be used with keys obtained from the visual password scheme we propose – however, it is important to note that conventional password schemes may be used as well, whether in combination, as a backup technique, or alone.

It should be noted that our focus in the design of jeu de paume is on solutions that are viable in the short-term, particularly for the current generation of palm devices. New technologies such as compact fingerprint readers, new forms of hardware tamper resistance, and the continuous Internet connectivity promised by 3G mobile phones may provide attractive alternative security mechanisms at some time in the future, such as that explored in [15, 16] for digital signing. We restrict our exploration to devices with limited or unreliable forms of connectivity and little effective tamper resistance, as is the case for most palm computers (and many mobile phones) today.

Organization

We describe each of the components of jeu de paume in successive sections. In section 2, we present the visual password system. We describe the credit-card vault in section 3, and funkspiel signatures in section 4. We have constructed PalmOS prototypes of the visual password system, and the credit-card vault. Where it is likely to be of interest to the reader, we discuss our experiences with these. We discuss background in the literature and in practice in the respective section for each security tool.

2 Visual Passwords

Text passwords have an extensive history in the world of computing, due in large part to the compactness of ASCII text and the universality of keyboards as data entry devices. The motivation for exploring alternative forms of authentication, however, is strong. Keyboards are no longer ubiquitous in computing devices, and numerous studies reveal both the ease with which users forget text passwords and their tendency to select passwords subject to dictionary attacks [11]. Computer memory and communication bandwidth, moreover, are available in ever increasing abundance.

A number of researchers have considered alternative forms of representing passwords. Psychological studies provide evidence, in particular, for superior retentiveness in visual memory over verbal memory in many human subjects. (A frequently cited example is the work of Shepard [20], which studies relative recognition memory capacity.) The literature on human memory and cognition

posits especially strong inate ability in human beings to recognize previously presented images [20, 23, 24]. In this vein, a commercial system called "Passfaces" [14] aims to exploit human users' well known capability for recognizing faces. In this system, the secret key of the user consists of a number of faces selected during an initialization phase. To authenticate herself, a user picks these initial faces from a challenge set containing a random lineup of spurious faces. A drawback of the "Passface" system is its need for a fixed database of images of faces; this database is cumbersome to store. In a system called "Déjà Vu", Dhammija and Perrig [9] seek to overcome this limitation by using randomly generated, abstract art images in lieu of faces in the challenge sets. In their system, the secret key of a user consists of a collection of such images.

While well conceived in the way that they draw on the cognitive strengths of users, both the Passfaces system and Déjà Vu are geared toward use with large screens, and not handheld devices. A serious problem in any setting, though, is their vulnerability to *intersection attacks*. In order that spurious images do not become familiarly recognizable to the user and therefore confused with the correct ones, it is necessary to rotate new sets of spurious images into challenge sets on a frequent basis. Thus an attacker viewing multiple challenges sets can in principle identify the correct images as those that are presented repeatedly, i.e., can identify correct images as the intersection of multiple challenge sets. Observing this difficulty, Dhammija and Perrig propose serveral possible remedies. One is to have the user change secret images on an ongoing basis, e.g., after every successful authentication. Dhammija and Perrig have not yet, however, studied the ability of users to handle such frequent changes in their target images.

In a somewhat different vein, Blonder [5] describes a system based on distinguished features in an image known as "touchpoints". For example, in the image of a horse, the tail, nose, mane, and hoofs might each consistute a touchpoint. The secret key of the user in this system consists of sequence of touchpoints. For example, a user might enter her password by indicating the rearmost hoof, then the nose, and finally the tail of the horse.

Jerymn *et al.* propose a "graphical password" system specially constructed for use with palm devices. In their system, the user sketches a picture, e.g., a house or abstract design, in order to authenticate herself. The graphical password system achieves a certain degree of error tolerance by deriving the associated secret key from the set of lines intersected in an implicit underlying grid. While Jerymn *et al.* make good arguments respecting the underlying entropy of user-selected keys in their system, they do not (and perhaps cannot) consider potential vulnerability to "pictionary attacks", i.e., the visual analog of dictionary attacks. Another drawback of the Jerymn *et al.* system is the nature of the error tolerance it offers: The grid system does not reflect the distortions, e.g., simple affine projections, naturally introduced by human users when they draw objects. Frykholm [10] offers some preliminary new ideas to address this latter problem.

Another visual password system is provided by Passlogix, Inc. as part of their "V-Go" platform. In this system, the user authenticates herself by manipulating

objects in a graphical window so as to simulate a real-world activity. For example, there are variants in which the user may select and combine ingredients so as to cook a meal or mix a cocktail. Data entry in the V-Go system is quite cumbercome. In consequence, the system enourages use of what are effectively short, low-entropy passwords. This weakness is aggravated by the fact that users select their own passwords, opening the system up to attacks based on ordinary user preferences, e.g., guessing common drinks.

Our visual password system: The visual password system in jeu de paume is most closely allied with the proposal of Blonder. Our system, however, does not make use of distinguished points. Instead, *any* point in an image can serve as part of the secret key of the user. The system narrows this wide field of possibilities in two ways. First, it offers a natural form of error tolerance in which the user can touch any point within a certain, predefined *region* around the correct one. Second, the system associates a distinct icon with each region in the image. When the user places the stylus on the image, the icon for the specified region is displayed. Hence, the user has two mnemonic stimuli guiding her toward the correct point: The objects composing the underlying image and the icons associated with regions.

We implemented our system to work on any of the current generation of PalmPilot models. On first interacting with the visual password system, the user is instructed to drag the stylus for several seconds in an arbitrary fashion across the screen in order to provide random seeding material. The user then selects her password. She is then presented with a sequence of k images, where k is a parameter based on the desired password entropy. On each image, two randomly generated points are presented. The user selects one of these two points as part of her password. Thus, in our system a balance is sought between user password selection, with its benefits of heightened password retention (see, e.g., [21]), and random password generation, which is beneficial for achieving good underlying entropy.

Regions in our system consist of 10×10 pixel blocks, each one of which, as explained below, has a unique associated icon. A designated point is centered in its block, so that the user can stray from this point anywhere in the region when entering her password. As there are 256 regions in total, each point/image pair in a visual password provides 8 bits of entropy. Thus, the full password has a total of $8k$ bits of entropy.[1]

We have implemented two versions of our system. The first, a "visual vault", involves selection of points on $k = 6$ images so as to produce a 48-bit visual password. This password serves as input to a key derivation function, namely

[1] This is true under the assumption that the user selects between the two alternative points in each image with equal probability during the initialization procedure. While this may or may not be the case, we believe that the impact on effective security of this choice is minimal. In the worst case, if user choice between points may be predicted completely by an attacker, the resulting effective entropy is reduced to seven bits per point/image instead of eight.

the PBKDF2 function from the PKCS #5 standard [18].[2] We calibrated this key derivation function so as to require roughly one second of compuation on the Palm V. This resulted in a parameterization using 500 iterations of the pseudorandom function. The Palm V has a 2.7 MIPS processor. Hence, we estimate the cost of breaking the system by brute force at over 12,000,000 MIPS years. Given that the peak speed attainable on a 1 GHz Pentium III processor is roughly 3000 MIPS [13], this should provide adequate security in most cases for the casual user.

We conducted an initial experiment designed to test the ease with which users were able to retain their memory of visual passwords. In a small pilot involving nine participants, we presented a 48-bit (i.e., $k = 6$) visual password and also a roughly 48-bit text password comprising 9 alphanumeric characters ('a'...'z', '0', ..., '9'); the former was selected through the visual password selection routine; the latter was selected uniformly at random[3]. Because not all participants owned PalmPilots, we referred them to a Web site that provided the passwords by means of applets, permitting participation in the experiment by means of PC-based browsers.

A week after password selection, participants were asked to re-enter the two forms of password. We measured a match in text passwords according to the size of the longest substring (not necessarily contiguous) present in both the original and the submitted password. Table 2 below indicates the percentage overlap according to this measurement, with a score of 0 recorded for refusal on the part of the user to enter a password (if, for example, the password was entirely forgotten). Also provided in Table 2 is the level of exact matching in the visual password according to the percentage of correctly selected points among the six images in the experiment. Finally, the last column of the table indicates the lapse of time between registration and password entry for each of the users. (Although users were asked to enter their passwords after a week, most took longer.)

These results are only preliminary. They are encouraging, however, in that, despite the larger number of successful authentications using text-based passwords, all users retained at least some memory of their visual passwords. It is clear nonetheless that greater refinement of our visual password system or more intensive user training is required to make the system wholly workable for a wide range of users. Smaller, more intensive user trials have yielded more promising results. We have found in particular that users who re-use their vaults on a daily basis for a short time after initialization are able to recall their visual passwords for long periods of time, and better than text-based passwords. Moreover, it

[2] Our prototype makes use of MD5 instead of the recommended HMAC-SHA1 as the underlying pseudorandom function. An updated version is to employ HMAC-SHA1 instead.

[3] While this is perhaps not the most even-handed basis for comparison, it is not clear how, for instance, random text password selection ought to be made interactive. Thus, we opted for a holistic system comparison as the only meaningful preliminary experiment for our idea.

User	*Text*	*Visual*	*Days*
1	0	50	8
2	0	33	7
3	0	16	10
4	44	16	14
5	0	66	8
6	0	83	9
7	0	16	7
8	100	83	8
9	100	100	8
Average	27	51	9

Table 1. Memory test: Experimental results showing the portions (in percentage terms) of correctly entered text password/visual password features

seems very likely that picture retention improves as users become familar with picture-based password schemes.

With the limitations of the stronger system in mind, a second variant of our system aims to achieve what might be described as PIN-level strength, i.e., entropy only slightly better than a four-digit PIN. This variant uses only two images, i.e., the setting $k = 2$. The relatively weak, 16-bit key provided by this system is useful in many applications. In one prototype variant, for example, we use it to furnish the PIN to a PalmPilot-based version of SecurID (in which PINs are typically four decimal digits in length, and thus around 13 bits). This variant may also be used in conjunction with the credit-card vault and funkspiel signature systems described below.

Future work: In response to user feedback, we are planning several enhancements to the visual password system. In the present version, background images are preselected. Psychological studies demonstrate better mnemonic retention, however, for objects that users have selected on the basis of preference in taste. Thus, we plan in a future version to offer a selection of "skins" among which the user might make selections for background images. Icons in the present system are more or less arbitrary. As a future enhancement, we plan to bind icons to regions on the basis of the content in background images. If a given region, for example, contains the top of a human figure's head, for example, the icon might be a hat, and so forth. Finally, our next version of the system will present images in randomized order during the training phase. This is done since the psychological literature suggests that memorization of data items in multiple, different contexts improves retention [6].

In summary, the visual password system in jeu de paume is specially geared toward use with handheld devices, for which it permits fast and easy password entry. In constrast to previous systems, the underlying entropy of passwords in our system may be precisely characterized; at the same time, the system promises good password (or short PIN) retention for a wide range of users.

3 Credit-Card Vault

The aim of our credit-card vault application is to provide good protection of credit-card numbers and similar information using *low entropy keys*. The underlying principle is simple: Instead of using a conventional form of encryption, we employ an encryption scheme that is *non-redundant* when applied to credit-card numbers. In other words, we create ciphertexts such that any decryption key yields what appears to be a valid ciphertext, i.e., credit-card number. Thus, this will cause analysis (by a powerful computer) to fail even though the the PIN is short, and even if the format of card numbers is known. Our technique can also be used on other short and random-looking data, such as PINs. (Thus, one could use one PIN or password – potentially visually based – to protect multiple other PINs and passwords, as well as credit cards.)

Example 1. To introduce the concept of non-redundant encryption, let us consider a simple example. Suppose that account numbers at Croessus Bank consist of ten random decimal digits, all of which are selected uniformly at random from the corresponding space. Let $A = a_1a_2 \dots a_{10}$ be such a number, where a_i stand for the i^{th} digit. Suppose further that the four-digit PIN $P = p_1p_2p_3p_4$ of a user is mapped by a function f to a unique stream of ten decimal digits. To produce a ciphertext C on the account number A, we first compute the stream $X = f(P) = x_1x_2 \dots x_{10}$. We then compute $C = c_1c_2 \dots c_{10}$, where $c_i = x_i \oplus a_i$; here $\oplus$ stands for digitwise addition mod 10. Thus, we offset the non-redundant part of the card (and only this) by XORing the output of a pseudo-random generator. Even though the seed to the latter (namely, the PIN) is short enough to be guessed by an attacker, the attacker will not be able to verify his guess. This holds since we only apply the offset to non-redundant parts. We assume here that the distribution of the non-redundant parts is uniform or nearly uniform in the view of an attacker. (This might not be the case for an attacker with "insider knowledge", but such an attacker can probably obtain credit card numbers using more direct means in any case.)

Observe that given C, an attacker cannot distinguish, even information theoretically, among the set of ten thousand possible corresponding plaintexts. In contrast, naïve use of, e.g., AES [2], for the encryption operation would reveal the account number to an attacker: It is very likely that all incorrect four-digit PINs would, on decryption of C, yield invalid credit-card numbers, e.g., numbers that are not strictly decimal in form.[4] Observe also that our scheme can be used to encrypt multiple account numbers simultaneously.

Non-redundant encryption is employed in many different forms in the cryptographic literature and practice. It may be viewed as the essential idea behind

[4] Assuming for the sake of example an 80-bit block size, ASCII encoding of an account number in a single block, and an ideal cipher, the probability that an invalid PIN yields a valid account number here is a little more than 8×10^{-17}. Thus the expected number of invalid PINs yield valid account numbers is a little more than 8×10^{-13}. This is an upper bound on the probability that even one invalid PIN does so.

the one-time pad, for instance [19]. In a more recent setting, non-redundant encryption serves as the underpinning of the EKE password-based authentication protocol and its successors. In the EKE protocol, a public key is encrypted under a password in a non-redundant form that prohibits an attacker from mounting a direct dictionary attack [4]. Another example of its use is in the authentication system in the VPN product of Arcot Systems [25], where non-redundant encryption is used to secure an authentication key. Other examples are numerous.

Our non-redundant encryption system for credit-card information consists of two parts: (1) A *decimal stream cipher*, i.e., a stream cipher that produces decimal-digit output instead of binary output and; (2) A *redundancy extractor*, i.e., a procedure for stripping away redundant information in a credit-card number (and reconstructing it during decryption).

Decimal stream cipher: As explained above, naïve use of a conventional encryption algorithm does not enable us to achieve the desired form of non-redunancy here. Instead, we employ a form of stream cipher, denoted by $DecStream$, whose output consists of a stream of decimal digits. Let P represent the password or PIN provided by the user, both expressed as ASCII bitstrings. Let $\|$ denote bitstring concatenation. Let $pad()$ denote the appending of '0' bits needed to ensure a string of 512-bit blocks. We employ the RC4 algorithm as the basis for our algorithm, parameterized for use with a 128-bit key and table size 256. Let $RC4(k, i)$ denote the i^{th} byte of output of this parameterization of RC4 on input key k. Our decimal stream cipher $DecStream$ may then be specified in terms of the following pseudocode, where $length$ is a specification of the desired length of the output stream.

function $DecStream(length,\ salt,\ P)$
    $k = MD5(pad(data \parallel P))$;
    WHILE $D < length$
        $i \leftarrow i + 1$;
        $c_i \leftarrow RC4(k, i)$;
        IF $c_i < 200$;
            $d_{2D} \leftarrow c_i \bmod 10$;
            $d_{2D+1} \leftarrow \lfloor c_i/20 \rfloor$;
            $D \leftarrow D + 2$;
    OUTPUT $d_1, d_2, \ldots, d_{length}$;

In our implementation, we employ the vault data as the salt for the cipher, so as to avoid the need for having the user provide an initial random seed. Obviously, stronger salt can be provided, according to taste. We use $DecStream$ to encrypt a decimal string in the obvious manner, based on addition mod 10, rather than addition mod 2. It may be seen that $DecStream$ benefits from the underlying strength of RC4, i.e., the hardness of digit prediction for $DecStream$ is reducible to that of bit prediction in RC4. (This notion can be formalized straightforwardly through characterization of the ciphers as pseudorandom generators [12].)

Credit-card redundancy extractor: For the purpose of our exposition, we focus on Mastercard account numbers, although our system accepts most major credit cards. A Mastercard account number A comprises sixteen digits. We write $A = a_1 a_2 \ldots a_{16}$ according to our notation above. Let $[a_i, a_j]$ consist of the string of decimal digits between a_i and a_j inclusive.

The first digit a_1 in any Mastercard account number is a '5'. The value of digit a_2 determines the set of digits identifying the issuing bank; in particular, the issuer identifier extends across the set of digits $[a_2, a_{3+a_2}]$. The last digit, a_{16}, is called a *check digit*. This is an error-correction digit derived from $a_1 a_2 \ldots a_{15}$ according to an apparently unpublished but publicly available algorithm [1]. We denote this function by g, and thus write $a_{16} = g([a_1, a_{15}])$. The set of digits $[a_7, a_{15}]$ is in all cases reserved as part of the account number of the holder. We treat this digit set as random, and thus as the non-redundant core of the credit-card number[5]. It is these digits ($[a_7, a_{15}]$) that we encrypt.

It is easy to see now that an encryption of account number A under password P consists of the fifteen-digit ciphertext $C = ([a_1, a_6], [a_7, a_{15}] \oplus DC(7, salt, P))$. To decrypt ciphertext C under password P' as account number A', we let $[a'_1, a'_8] = [c_1, c_8]$, and $[a'_7, a'_{15}] = [c_7, c_{15}] \oplus DC(7, salt, P)$, and finally $a'_{16} = g([a_1, a_{15}])$. The same principle can be extended to the encryption of other random data elements with pre-determined structure and, as noted above, it is possible to encrypt multiple data elements simultaneously under the same password. Our credit-card vault is particularly convenient when used in conjunction with the two-slide version of our visual password system.

4 Funkspiel Signatures

4.1 Background: Funkspiel schemes

Funkspiel schemes are an idea recently introduced by Håstad *et al.* [22].[6] In conventional hardware security modules with continuous power sources, the usual strategy for providing tamper-resistance is known as "zeroization". When a breach is detected in the physical integrity of the module, the secret keys are erased. The idea behind funkspiel schemes is different. Rather than erasing secret keys, a funkspiel scheme modifies them in a way that is undetectable to the attacker, at least in a cryptographic sense. The hope is that an attacker who breaks into the security module will be unable to detect whether a change has occurred in the keys she recovers. Thus she will either be deterred from using these keys, or will unwittingly use invalid ones. If the attacker does make use of invalid keys, then a funkspiel scheme creates a provision for her behavior to be

[5] Given knowledge of bank policy, an attacker might be able to exploit information about account number assignment, but we consider this a minor risk.

[6] The term "funkspiel" may be translated loosely as "radio game" in German. The term describes a form of attack on underground radio networks during WWII that served as inspiration for the cryptographic construction. See [22] for more details.

detected and traced. In a sense, a funkspiel scheme involves the triggering of a "silent alarm". As such, it could provide security against theft and loss of mobile devices, in the sense that it would limit access to resources accessed from the device, thereby limiting the loss to the cost of replacement.

An alternative to detecting unauthorized access, and communicating this by means of silent alerts, is to attempt to prevent unauthorized access in the first place. While we have shown this to be possible using a PIN for short segments of random data (such as other PINs and credit card numbers), one cannot secure all the palm contents in this manner. This becomes possible using techniques in which more entropy is extracted and used as encryption and decryption key, such as the biometric protection techniques of Cloakware^{TM} [7]. Not considering aspects relating to the false-positive and false-negative rates of biometric methods, it is clear that these do not offer protection against an attacker that forces the device owenr to collaborate – in contrast, funkspiel schemes do. This holds since the device will work using a wide range of PINs (if not all), but all PINs but the correct will sound a silent alarm – silent because it cannot be detected by the attacker, even if he is given full access to the device and its data, and to old transcripts of the device. The only party that could detect the alarm is a so-called "monitoring center". This monitoring center knows the PINs of all users it monitors, and verifies that the correct PINs were used, as it obtains signature transcripts. The monitoring center could therefore reside either with a certification authority that is contacted by verifiers wanting to verify that the certificate in question still is valid, or with a service provider, such as the user's bank, were the signatures to be used for funds transfers. A signature generated using the secret key stored on the device, but using the wrong PIN, would therefore be distinguishable from an "entirely correct" signature, but only to the monitoring center, and not to any other party (including the attacker). This holds even if the attacker has access to old transcripts.

Funkspiel schemes were originally designed to help secure devices implementing authentication protocols. For example, a (powered) smartcard might apply a MAC (message authentication code) under secret key k to each transaction it performs; this would enable transactions to be validated by a bank server. One might try implementing a funkspiel scheme as follows. On detecting a physical security breach by an attacker, the smartcard replaces the key k with a false key $k' = h(k)$, for some hash function h. In contrast, we do not assume any tamperproofness, but use PINs to "secretly validate" signatures, and incorrect PINs to signal theft and distress.

In a traditional funkspiel scheme, an attacker would be unable to detect the change in MAC key by examining the contents of the smartcard alone, and might be tempted to create a fake smartcard using the MAC key k'. The bank would be able to detect this, and indeed verify that the funkspiel scheme was triggered, since the bank knows k, and can therefore determine k'. On the other hand, even if the attacker knows that a funkspiel scheme is being used, she cannot tell whether the key she recovers is correct, and cannot compute k even if she obtains k' and suspects that it is a false key. (In our scheme, instead, the attacker cannot

determine whether a given PIN is valid or not, even when given full access to the contents of the corersponding device, and to previous transcripts.)

The problem with the above naïve approach is that the "silent alarm", i.e., the change from the original key k to the false key k', is not truly silent. If the attacker can view transcripts produced by the smartcard prior to break-in, then she can examine the associated MACs and determine whether the key she recovers is valid or not. The funkspiel schemes in [22] are designed to address this problem, and are undetectable even to attacker with access to all of the communications of the protected hardware device.

The inability of the attacker to detect a key swap in a funkspiel scheme is referred to as the *stealth* property. This property is formally modelled by the following game. Alice holds a secret key k. She sends derivative keys $y_1, y_2, \ldots$ to Bob for some number of timesteps. The attacker, Eve, is able to see these y_i values, and adaptively decides when she wishes to break into Alice's key storage. Prior to the break-in, Alice flips a coin, obtaining bit b. If $b = 0$, then she leaves the key k unaltered. Otherwise, she is permitted to replace k with some false key k' – in our setting, we consider different PINs instead of keys. Eve then sees the key held by Alice. Eve's task is to guess the bit b. If she can do so with a non-negligible advantage, i.e., probability non-negligibly greater than $1/2$, then she wins the game. Otherwise the funkspiel scheme is regarded as having the stealth property. Another important property for a funkspiel scheme is that of *unforgeability*. Stated briefly, Eve should be unable with non-negligible probability to cause Bob to accept a new y_i value that she herself produces.

4.2 Funkspiel signatures

Håstad *et al.* present three different funkspiel schemes, each with different properties. We show how to adapt one of these schemes for protection of a private digital signing key. The setting we consider, however, is rather different from that envisaged by the Håstad *et al.* As we have explained, we do not assume any kind of tamper resistance in the palm devices we are considering in this paper. Thus, there is no way to detect an attack in an active way in a palm device; stated another way, there is no physical "tripwire" for the funkspiel mechanism. Instead, we show how to employ the funkspiel idea so as to protect a digital signing key using a PIN known to the user. This PIN is not present in the device and, thanks to the security properties of the funkspiel scheme, cannot be extracted from digital signatures produced by the palm device. Thus, we can construct a system in which an attacker may be viewed as triggering the funkspiel mechanism by incorrectly guessing the PIN of the user. We refer to signatures produced this way as *funkspiel signatures*.

A funkspiel signature scheme involves three types of participant: A *signer*, a *verifier*, and an entity that we refer to as a *monitoring center*. The signer and verifier operate essentially as in a conventional digital signature scheme. The signer produces digital signatures using a private signing key, but also using a PIN, which she furnishes at the time of signing and incorporates into her signatures in a special way. Use of funkspiel signatures results in no noticible

difference for the verifier. Indeed, the verifier need not even be able to distinguish a funkspiel-enabled digital signature scheme from a conventional one. The monitoring center may be viewed as a type of second-line verifier, whose aim is to check that the signer employed the correct PIN at the time of signing. Only the monitoring center should be able to check the PIN. In particular, an attacker – or even an ordinary verifier – should not learn any information about the PIN associated with a digital signature. The security goal of the scheme, then is this. An attacker who steals the palm devices of an honest user is unlikely to be able to guess the PIN of the user, and cannot learn information about the PIN from correctly issued digital signatures. Thus, even though the attacker may gain access to the private signing key in the palm device, and can produce signatures that look correct to the verifier, the attacker is likely to produce signatures using the wrong PIN, thereby alerting the monitoring center. In a typical scenario, the signer might be a consumer, the verifier a merchant, and the monitoring center a bank. The bank, on seeing invalid PINs, would be able to close the associated user account, or take other action according to its chosen policy.

We take the view that it is best to employ an existing digital signature scheme and standard for the purpose of achieving the funkspiel property, rather than to develop a new one. In particular, we show how to incorporate the funkspiel property into PSS RSA signatures [3]. PSS is short for *Probabilistic Signature Scheme*, and is a padding method of RSA developed by Bellare and Rogaway. Its aim is to yield a digital signature scheme whose security is reducible to standard cryptographic assumptions, namely the standard RSA assumption and the random oracle assumption on a hash function.

PSS is an IEEE P1363a standards submission (likely to be approved this year), and is part of PKCS #1 v2.1, which is in its final 30-day review at the time of writing. The message recovery version of PSS is also in ISO/IEC 9796-2, and is likely to be approved during the next few months. Thus, it is likely to see wide use in coming years.

How PSS works. In order to sign a message M, a signer first pads it using a random pad r, and the technique proposed in [3], which we detail below. The padded message is then signed, using the standard RSA signature scheme, and the message and corresponding pad communicated along with the signature. A verifier again pads the message according to the PSS scheme, after which the signature on the resulting string is verified. We will use the very same structure, with a minor modification: We let the random pad r be a pseudo-random function of both a seed *and* a PIN p, instead of only of the seed. An attacker will not be able to distinguish a pad from the correct PIN from a pad generated from the incorrect PIN. However, the monitoring center, who knows the user's PIN, can distinguish between the two cases. Let us now consider in detail how PSS works:

1. First, a random string r is selected uniformly at random from $\{0,1\}^{k_0}$ for some security parameter k_0, of proposed size $k_0 = 128$.

2. Second, $hash(M|r)$ is computed, where M is the message to be signed; $hash : \{0,1\}^* \rightarrow \{0,1\}^{k_1}$ is proposed to be implemented using MD5 [17]; and where $|$ denotes concatenation. The result is denoted w.
3. Third, the value $g_1(w)$ is computed, where $g_1(w)$ returns the first k_0 bits of $g(w)$, and where $g : \{0,1\}^{k_1} \rightarrow \{0,1\}^{k-k_1-1}$, again, suggested to be implemented using MD5.
4. Next, r^* is computed as $r^* = g_1(w) \oplus r$.
5. Finally, $g_2(w)$ is computed, where $g_2(w)$ returns the last $k - k_0 - k_1 - 1$ bits of $g(w)$, as defined above. The output is the string $(0|w|r^*|g_2(w))$.

Adding Funkspiel Capabilities. As mentioned, instead of letting r be the output of a pseudo-random function that takes a seed and a counter as its only inputs, we obtain it as a pseudo-random function of a seed; a counter; and the user-entered PIN p. Thus, the value r, used for padding, communicates the PIN entered to the monitoring center, who can determine if the correct PIN was used (since it knows the PIN of each user registered with it.) As explained above, however, an attacker should not be able to distinguish a "good PIN" pad from a "bad PIN" pad – even given previous transcripts. In order to achieve this, the following technique is employed:

Setup. The signer and the monitoring center select a random seed σ_0 that is kept secret from all others, but stored both with the monitoring central and the user device. In addition, the user selects a PIN p, that is given to the monitoring central, and stored by the same. A counter cnt is set to zero.

Generation of pad. The user device prompts the user for a PIN p, and computes

$$\begin{cases} cnt \leftarrow cnt + 1 \\ \sigma_{cnt} \leftarrow h_1(\sigma_{cnt-1}) \\ r \leftarrow h_2(\sigma_{cnt}, p). \end{cases}$$

Here, h_1 and h_2 are one-way functions, and may be implemented using MD5. The value r is used in the PSS signature generation, as described above. In the above and in the following, we do not indicate the time (or invocation number) in the variables, in order to simplify the notation.

Verification of funkspiel signature. A PSS funkspiel signature, generated using PSS for a random pad r generated as above, is verified as standard PSS signature. Thus, the standard verifier pays no attention to the shape of the value r, as long as it conforms to the requirements outlined by PSS.

Verification of pad. The monitoring center verifies the correctness of the pad r used in a PSS signature after verifying the signature itself. If the signature is valid but the "wrong" value of r is used, then it knows that the user must have entered the wrong PIN p. Depending on its policy, it may act on this in various

ways, as will be described below. The verification is performed by computing

$$\begin{cases} cnt \leftarrow cnt + 1 \\ \sigma_{cnt} \leftarrow h_1(\sigma_{cnt-1}) \\ \tilde{r} \leftarrow h_2(\sigma_{cnt}, p), \end{cases}$$

checking whether $r = \tilde{r}$. (Recall that the monitoring center stores (σ_{cnt}, cnt, p) for each user – the appropriate entry is selected given the public key of the signature.)

Handling user errors. The above technique allows for a variety of responses of the monitoring center, were an incorrect PIN to be used. For example, the user device may be blacklisted; the user may be contacted; or a more complex reaction may be performed, potentially based on several received transcripts, and depending on the nature of the messages signed. However, in order to avoid such costly measures to be taken each time an honest user enters the wrong PIN, a technique like the following may be employed:

As above, the user PIN is p. We let π be the result of applying a function F to p; here, the entropy (size) of π should be much lower that that of p. For example, if p is a four-digit PIN, π may be a check-digit obtained by adding all the digits of p, modulo 10. The value π would be stored on the user device. Each time a user enters his PIN p, the device compares $F(p)$ to π, and alerts the user if these are different. Note that while this gives some information about p to an attacker, it does not allow the attacker to determine the value of p, even with access to previously produced transcripts. However, it catches a large portion of incorrect PINs, making it less likely that an incorrect PIN corresponds to a mistake by the user.

Protection against duress. We note that the above error detection feature does not make the resulting technique susceptible to attacks in which the attacker forces the device owner to give him the PIN. This is so since it is so easy to modify the PIN that it can be done even under duress. In particular, the threatened user can permute the digits in the PIN as he tells it to the attacker. While this reduces the security substantially, compared to having the full entropy of PINs, it is better than nothing. A more sophisticated (or perhaps: paranoid) device owner may instead memorize and give out an "emergency PIN" with the correct checksum.

Synchronization issues. The verification performed by the monitoring center relies on synchronization between it and the signer. However, it is evident that signatures may be processed and verified out of order, requiring means for these two parties to synchronize. Thus, instead of generating r as above, the signer may generate it as follows:

$$\begin{cases} cnt \leftarrow cnt + 1 \\ \sigma_{cnt} \leftarrow h_1(\sigma_{cnt-1}) \\ r' \leftarrow h_2(\sigma, p) \\ r \leftarrow E_K(cnt, r') \end{cases}$$

Here, K is a shared key between the user and the monitoring center (each user gets assigned a unique such key), E corresponds to symmetric encryption, e.g., using AES [2]. As the monitoring center receives a PSS signature using a padding of the above format, he looks up the value of K (given the public key associated with the signature); computes cnt by decrypting r, and uses this to syncronize to obtain the appropriate value of σ and r'. He then verifies whether r' was computed using the correct PIN p. Since the monitoring center will verify signatures out of order, it will have to store an old value of the seed, such as σ_0, and compute newer values of the same from this. To reduce the amount of storage and computation needed for this, a technique such as [8] may be employed.

Efficiency. The extra operations required in order for us to add the funkspiel property to PSS signatures do not result in a large computational overhead for neither signer nor monitoring center (and nothing at all to a standard verifier, who cannot detect the modification.) In particular, both signer and monitoring center need to evaluate a hash function twice, and perform one symmetric encryption (or decryption). In addition, the monitoring center needs to compute the value σ_{cnt-1} employed by the signer from some stored seed – this requires a number of hash function evaluations corresponding to the distance to the seed (or other stored intermediary value). This value will be relatively small if signatures are verified (by the monitoring center) in an order largely corresponding to that in which they were generated. Our modification also does not demand a large quantity of extra storage, but merely enough to store the above mentioned values. Therefore, our modifications do not result in a noticable increase in neither computation nor storage, compared to "plain" PSS signatures.

Security Analysis. In order to state our security claims, we consider the following game, adapted from [22]. First, the attacker A may be characterized by a pair of algorithms A_{intr} and A_{guess}, as follows.

- An adaptive intrusion algorithm A_{intr}. Input to the algorithm is a sequence of messages and corresponding signatures $(m_1, s_1), (m_2, s_2), \ldots, (m_i, s_i)$ for $i \geq 0$. For $i = 0$, we denote the sequence by ϕ. The output of A_{intr} is a message m_{i+1} or the special message "break".
- A guessing algorithm A_{guess}. A_{guess} takes as input a secret σ_{i+1} and a message/signature sequence $(m_1, s_1), (m_2, s_2), \ldots, (m_i, s_i)$. For $i = 0$, this may be regarded as a null sequence. The output of A_{guess} is '1' if the algorithm has reset the PIN, and '0' otherwise.

Given a funkspiel scheme FS and an attacker A, the experiment now is as follows:

Experiment $STEALTH(\mathsf{FS}, A)$
$s_1 \leftarrow \mathsf{KeySet}(k)$
$p \leftarrow \mathsf{PIN} - \mathsf{reset}$
$i \leftarrow 1$

$m_1 \leftarrow A_{\mathsf{Intr}}(\phi)$
while $m_i \neq$ "break" do
    $s_i \leftarrow \mathsf{Sig}(m_i, \sigma_i, p)$
    $i \leftarrow i + 1$
    $\sigma_i \leftarrow \mathsf{KeyEv}(\sigma_{i-1})$
    $m_i \leftarrow A_{\mathsf{Intr}}((m_1, s_1), \ldots, (m_{i-1}, s_{i-1}))$
select $y \in_u \{0, 1\}$
if $(y = 1)$ then
    $p \leftarrow \mathsf{PIN} - \mathsf{reset}$
$g \leftarrow A_{\mathsf{guess}}(s_i, (m_1, c_1), \ldots, (m_{i-1}, c_{i-1}))$
if $g = y$ then
    return '1'
else
    return '0'

In the above, $\mathsf{PIN} - \mathsf{reset}$ is a function that selects the pin p uniformly at random from the appropriate range.

Definition 1. *Let* FS *be a funkspiel signature scheme with security parameters j, k and l. The scheme has the stealth security property if for any adversary A with resources polynomially bounded in security parameters k and l, it is the case that $|\frac{1}{2} - STEALTH(\mathsf{FAC}, A)|$ is negligible, i.e., asymptotically smaller than any polynomial in j, k, and l.*

With this definition in place, we state the following rough claim.

Claim. Our proposed funkspiel PSS scheme has the stealth security property under a reduction to the following two assumptions: (1) The random oracle assumption on the underlying hash functions and (2) The assumption that the encryption function is computationally secure against adaptive chosen ciphertext attack.

We note that a variant of PSS, allowing message recovery, was also proposed in [3]. This variant, referred to as PSS-R, may be augmented for funkspiel in a manner similar to what we have described above.

5 Conclusion

The three components of jeu de paume are naturally complementary. The visual password scheme provides a way for users to enter short keys conveniently on palm devices. The credit-card vault and funkspiel signature scheme represent ways of using such keys for secure operation despite: (1) The fact that these keys are too short to provide a cryptographic strength of security and (2) Unreliable network connectivity. The credit-card vault achieves a special form of data encryption, while the funkspiel scheme helps protect digital signing keys in such constrainted environments.

We conclude by noting that there are other resources exploitable by the security architect that we have not explored in this paper. For example, it may

prove interesting to explore the possibility of periodic, unreliable network connections as a mechanism for security enhancement. Another possibility is that of helping to secure palm devices using small push-button devices, such as the key fobs used to unlock automobiles. Given the comfort of consumers with such fobs, they might well prove a promising ancillary security tool.

Acknowledgments

We offer thanks to Debbie Stolper and Susanne Wetzel for their ideas and comments on this work.

References

1. ANSI standard X4.13-1983. American National Standards Institute, 1983.
2. Advanced encryption standard, 2001. FIPS Standard FIPS-197.
3. M. Bellare and P. Rogaway. The exact security of digital signatures: How to sign with rsa and rabin. In U. Maurer, editor, *Advances in Cryptology - EUROCRYPT 96*, pages 399–416. Springer-Verlag, 1996. Lecture Notes in Computer Science Vol. 1070.
4. S. M. Bellovin and M. Merritt. Encrypted key exchange: Password- based protocols secure against dictionary attacks. In *Proceedings of the I.E.E.E. Symposium on Research in Security and Privacy*, Oakland, 1992.
5. G. Blonder. Graphical passwords, 1996. United States Patent Number 5559961.
6. L. Cermak. *Improving Your Memory*. McGraw-Hill, 1976.
7. Cloakware, 2002. URL: www.cloakware.com.
8. D. Coppersmith and M. Jakobsson. Almost optimal hash sequence traversal. In *Financial Cryptography*, pages ???–??? Springer Verlag, 2002.
9. R. Dhammija and A. Perrig. Déjà Vu: A user study, using images for authentication. In *Proceedings of the 9th USENIX Security Symposium*, pages 40–46, 2000.
10. N. Frykholm. Passwords: Beyond the terminal interaction model. Master's thesis, Umeå University, Department of Computer Science, 2000.
11. D. V. Klein. Foiling the cracker: A survey of and improvements to, password security. In *UNIX Security II: USENIX Workshop Proceedings*, pages 5–14, Berkeley, CA, 1990.
12. M. Luby. *Pseudorandomness and Cryptographic Applications*. Princeton University Press, 1996.
13. D. Mihocka. Pentium 4: In depth, 2001. URL: http://www.emulators.com/pentium4.htm#Analyzing.
14. Passfaces, 2002. URL: www.passfaces.com.
15. M.K. Reiter and P. Mackenzie. Delegation of cryptographic servers for capture-resilient devices. In P. Samarati, editor, *ACM CCS '01*, pages 10–19, 2001.
16. M.K. Reiter and P. Mackenzie. Networked cryptographic devices resilient to capture. In *IEEE Security and Privacy*, pages 12–25, 2001.
17. R. L. Rivest. *The MD5 Message-Digest Algorithm*. RFC 1321, Internet Activities Board, 1992.
18. RSA Laboratories. PKCS #5: Password-based cryptography standard 2.0, March 1999. http://www.rsasecurity.com/rsalabs/pkcs/pkcs-5/index.html.

19. C.E. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28(4):656–715, 1948.
20. R. N. Shepard. Recognition memory for words, sentences and pictures. *Journal of Verbal Learning and Verbal Behavior*, 6:156–163, 1967.
21. N. J. Slamecka and P. Graf. The generation effect: Delineation of the phenomenon. *Journal of Experimental Psychology*, 16:272–279, 1978.
22. J. Håstad, J. Jonsson, A. Juels, and M. Yung. Funkspiel schemes: An alternative to conventional tamper resistance. In S. Jajodia, editor, *Seventh ACM Conference on Computer and Communications Security*, pages 125–133. ACM Press, 2000.
23. L. Standing. Learning 10,000 pictures. *Quarterly Journal of Experimental Psychology*, 25:207–222, 1973.
24. L. Standing, J. Conezio, and R.N. Haber. Perception and memory for pictures: Single-trial learning of 2500 visual stimuli. *Psychonomic Science*, 19:73–74, 1970.
25. Arcot Systems, 2002. URL: www.arcot.com.