Funkspiel Schemes: An Alternative to Conventional Tamper Resistance

Johan Håstad
Royal Institute of Technology*
Stockholm, Sweden
johanh@nada.kth.se

Jakob Jonsson
RSA Laboratories
Stockholm, Sweden
jjonsson@rsasecurity.com

Ari Juels
RSA Laboratories
Bedford, MA, USA
ajuels@rsasecurity.com

Moti Yung
CertCo, Inc.
New York, NY, USA
moti@certco.com
moti@cs.columbia.edu

September 25, 2013

Abstract

We investigate a simple method of fraud management for secure devices that may serve as an alternative or complement to conventional hardware-based tamper resistance. Under normal operating conditions in our scheme, a secure device includes an authentication code in its communications, e.g., in the digital signatures it issues. This code may be verified by a fraud management center under a pre-determined key σ. When the device detects an attempted break-in, it modifies σ. This results in a change to the authentication codes issued by the device such that the fraud management center can detect the apparent break-in. Hence, in contrast to the case with typical tamper-resistance schemes, the deployer of our proposed scheme seeks to trace break-ins, rather than prevent them. In reference to the wartime practice of physically capturing and subverting underground radio transmitters – a practice analogous to the capture and use of secret information on secure devices – we denote this idea by the German term *funkspiel*, meaning "radio game."

One challenge in constructing a funkspiel scheme is to ensure that an attacker privy to the authentication codes of the secure device both before and after the break-in, as well as the secrets of the device following the break-in, cannot detect the alteration to σ. Additional challenges

*Some of this work was done while visiting RSA Laboratories.

involve minimizing the communication and computation overhead, the requirement for use of shared secrets, and the state information associated with the authentication codes. We present several simple and practical schemes in this paper.

1 Introduction

In 1942, a member of the underground Allied radio network in Holland named Lauwers was captured by German intelligence agents. Lauwers had been trained in such an event to feign cooperation with his captors, but to alert London covertly by omitting a security check from his messages. For Lauwers, the security check consisted of a deliberate corruption of every sixteenth letter in his messages. The Germans, however, were aware of the existence of such security checks, and also in possession of three messages previously transmitted by Lauwers. Hoping to be able to transmit further, misleading messages to the Allied underground, the Germans demanded that Lauwers reveal his security check. As it happened, the sixteenth letter in two of the three captured transmissions was the letter 'o' in the word "stop". Lauwers, therefore, cleverly claimed that his security check consisted of a periodic corruption of the word "stop". This claim was seemingly consistent with previous transmissions, and successfully spoofed Lauwer's German captors.[1] The Germans referred to the practice of subverting radio transmission setups like that of Lauwers as a *funkspiel*, or "radio game" [17].

This historical anecdote mirrors a scenario encountered in contemporary security systems. The underground transmitter may be regarded as analogous in function to a secure hardware device, such as a smartcard, while the intelligence agents represent an attacker attempting to break into the device. In this case, the "funkspiel" mounted by the attacker is an attempt to compromise the secrets contained in the device in such a way that the secrets remain valid.

While manufacturers of secure devices have developed a panoply of countermeasures against invasive attacks, these devices remain largely vulnerable to a variety of probing techniques [3, 4, 9, 18]. One of the most effective measures for protecting tamper resistant modules is to have them "zeroize", that is, obliterate, sensitive information when a break-in attempt is detected. In this way, a compromised device is disabled, and therefore rendered valueless to an attacker. It is often possible for a sophisticated attacker to circumvent such countermeasures, and thus capture intact the secret information in the device [3]. The example of Lauwers offers an alternate strategy. Rather than zeroizing sensitive information when a break-in is detected, we can instead alter it. In this way, a forger will have greater difficulty telling whether or not his attempts to circumvent the countermeasures in the card have been successful. If the al-

[1] The cleverness of Lauwers was not ultimately rewarded. The Allies, accustomed to poor telegraphy skills in their agents, assumed that the incorrect security check was a mistake.

teration of sensitive information goes undetected, then a system administrator can be covertly alerted to the compromise of a device. Instead of seeking to prevent device compromise in this case, the system administrator is then able to identify and trace successful attacks. We refer to this approach as a *funkspiel scheme*.[2] Funkspiel schemes may serve as an alternative to the approach of zeroizing secrets, or else as a complementary technique.

In its most naive form, a funkspiel scheme might be as follows. A smartcard generates signatures using private key SK under normal operating conditions, but replaces this key with a new, unrelated key SK' upon detection of a break-in attempt. The problem with this approach is that an attacker with access to signatures previously generated by the card will be able to detect the change in key, and thereby determine that the device is seeking to transmit a covert alarm.

The trick of Lauwers, of course, was to alter his authentication check in such a way that it was seemingly consistent with the transcripts of his past transmissions, but capable of alerting an allied party. In this paper, we show how to achieve an analogous property in a cryptographic setting. This provides us with a new approach for managing fraud in a system dependent upon tamper resistant devices. In a smartcard-based ATM cash withdrawal system that uses conventional tamper-resistance methods alone, a forger attempting to break into a smartcard will find himself either capable of defeating the tamper-resistance measures, yielding a false cash card without detection, or else with a damaged and non-functioning card. Given the use of a funkspiel scheme in such a system, there is an intermediate possibility. The forger may trigger a covert alarm, providing the system administrator with an opportunity to identify and trace forged cards, and ultimately locate the forger.

1.1 Organization

In section 2, we review previous, related ideas in the cryptographic literature. We propose three different funkspiel schemes in section 3. We discuss security in section 4, and conclude in section 5 with proposals for some directions for future research.

2 Previous Work

The covert channel through which a device alerts a system administrator to a compromise in a funkspiel scheme is similar in flavor to a *subliminal channel*. The notion of a subliminal channel arises in the context of the *prisoners' problem*. Here, two players, known as "prisoners", agree in advance upon a secret key σ. Their aim then is to hide information in seemingly innocuous messages

[2] We reject the perhaps somewhat more accurate term "anti-funkspiel scheme" as too cumbersome and confusing.

in such a way that a third player with access to their communications, known as a "jailor", cannot read or even detect the hidden information. Simmons [23], for example, investigated methods of superimposing covert channels on DSS signatures. These schemes are such that even if the jailor captures the signing keys of the prisoners, he cannot detect the existence of the subliminal channel. Capture by the jailor of σ, however, will uncover the contents of the channel.

Another closely related notion is that of *deniable encryption*, as proposed in [11]. Deniable encryption is an asymmetric encryption scheme in which the sender encrypts a plaintext m as ciphertext c under a randomization factor r. The receiver can successfully extract the plaintext m. The sender is capable, however, of determining a different pair (m', r') such that $m' \neq m$, but c appears to an attacker to be a correct corresponding ciphertext. The aim here is for the sender of a message, placed under duress by a third party, to be able to repudiate the originally transmitted plaintext. The idea of producing false plaintexts consistent with previous outputs is similar to the goal of a funkspiel scheme. In general in a funkspiel scheme, however, the information of the sender at the time of capture need only be such that the attacker cannot detect inconsistency with previous transcripts. There is no need, as in a deniable encryption scheme, for the sender to produce plaintexts or keys for these transcripts. (We do nonetheless propose one funkspiel variant meeting this requirement.) At the same time, while deniable encryption seeks to protect previously transmitted messages, a funkspiel scheme in essence aims to corrupt future messages in an undetectable fashion. In the same spirit as denable encryption, deniable authentication seeks to enable repudiation of previously authenticated messages. This idea is addressed in [13, 5].

In an electronic secret-ballot election, a voter must be able to prove securely to the voting authorities that she cast her vote in a particular way. At the same time, so as to avoid the possibility of vote-buying or forcible coercion of voters, it is desirable for a voter to be unable to prove her choice to another party. Schemes with this latter property, investigated in, e.g., [7, 22, 21], are referred to as *receipt-free* voting schemes. As a ciphertext ballot cast by a given voter is generally visible to other voters, and thus to potentially malicious parties, the aim in a receipt-free voting system is much like that in a deniable encryption scheme. The voter must be able to produce two different plaintexts for a given ciphertext ballot, so that it is impossible for the voter to construct a proof that she has voted in a given fashion. As it is possible for a vote-buyer to construct the ciphertexts in advance on behalf of the voter, deniable encryption does not quite accomplish the aims of a receipt-free voting system, and it is not known how to construct a correct receipt-free voting scheme without the aid of untappable channels. Thus the framework in receipt-free voting systems is rather different from that which we consider here.

As we do in this paper, Burmester, Desmedt, and Seberry [10] as well as Bellare and Miner [6] consider the notion of protecting past history in a device whose secret information has been compromised. They propose what they refer

to as a *forward-secure* signature scheme. The signer in their scheme holds a fixed public key PK and an initial private key SK_1. On signing the i^{th} message, the signer discards the key SK_i and generates a new private key SK_{i+1}. Thus, the Bellare and Miner scheme employs a mechanism for key evolution after each step as a means of protecting past information. The special security property of the Bellare and Miner scheme is that an attacker who has seized key SK_i is unable to forge messages with indices less than i. He is, however, able to forge future messages straightforwardly. The scheme in [10] provides forward security to escrowed keys for a public-key cryptographic systes. Once a key is removed from escrow, associated past messages can no longer be opened.

Like the above schemes, our constructions in this paper draw on techniques of key evolution as a mechanism for protecting past history. In contrast to the scenario envisaged in [6], however, we consider that an attack may be detected by the device protecting the signature key of the user. Our goal is therefore somewhat different. The Bellare and Miner construction employs alteration of signature secrets to prevent attacks involving expired signature keys. We impose the additional requirement, when an attack is detected, that alteration of signature secrets be undetectable by the attacker, even with access to previous public transcripts. Given our exploration of a different security model, we are able to achieve wider variety and greater efficiency in our constructions than [6]. Nonetheless, our techniques and goals are somewhat similar in flavor, and [6] represents important antecedent work.

The notion of a covert distress channel achieved by alteration to a secret key has appeared previously in a proposal for "distress cash" by Davida *et al.* [12]. They describe a cash system in which each user possesses two PINs, one PIN for normal use and a secondary "distress" PIN. When placed under duress, e.g., when physically coerced, the user employs the distress PIN, rather than the normal one. With use of the distress PIN, the transaction proceeds in an apparently normal fashion, but a covert distress signal is transmitted to the appropriate authorities.

A distress PIN relies on the sustained integrity of the transaction transcripts of the system in which it is used. An attacker with access to transcripts of previous authentication sessions and capable of breaking into the device of the sender can distinguish a valid PIN from the duress PIN. As explained above, our goal in this paper is similar in spirit to that motivating deployment of distress PINs, except that we seek to employ hardware modification of secrets, rather than human modification, and aim to tolerate compromise of hardware integrity.

2.1 Example funkspiel deployment

To provide a more concrete idea of how a funkspiel scheme might be deployed, we briefly sketch an example. Consider an ATM (Automatic Teller Machine) system in which a user U possesses a smartcard containing a private signature key SK_U and a certificate C_U on the corresponding public key. When Alice wishes

to make a withdrawal, she inserts her card into the ATM machine. By authenticating herself in an appropriate fashion to the card using, e.g., a PIN, she produces a signature $\Sigma_{SK_A}[m]$ on some message $m =$ "I want to withdraw \$50". The ATM verifies the validity of m prior to dispensing the requested cash. The ATM also verifies Alice's account balance and other information by communicating with a central bank server.

The integrity of the cards in this system depends upon the inability of an attacker to seize SK_U from the card of a legitimate user U or to modify the card so as to produce signatures without proper authorization. To protect the system using a funkspiel scheme, we might include an additional secret s_U in the card of user U, and also store s_U in the record for U in the central bank server. In a given transaction originally involving message m, the card for user U in this modified system appends to m a corresponding code c_m based on s_U. The validity of this code is verified by the central server during the cash withdrawal.

If an attacker breaks into a card in a detectable fashion, the card replaces the secret s_U (or a derivative thereof) in an undetectable manner with some secret s'_U. When the attacker attempts to use the card at an ATM, producing signed message m, the corresponding code c_m will be incorrect. The ATM will thus learn of the fraud attempt on communicating with the central server. Appropriate action may then be taken to assist fraud management personnel: a silent alarm may be triggered or the user may be photographed. The conventional tamper resistance mechanism, namely erasure of card secrets, would instead render a card invalid, alerting the attacker to her failed attempt, and enabling her to avoid being traced or captured.

3 Several Funkspiel Schemes

We may regard a funkspiel scheme as including the participation of three players, a *sender*, a *receiver*, and an *attacker*. In an initial keying step, the sender and receiver compute a shared secret key using an algorithm KeySet, or else exchange certified public keys. In time step i, the sender is presented with a message m_i, typically selected by the receiver or generated in response to some desired transaction. In an attack scenario, it is possible that m_i may be selected by the attacker. The sender employs an algorithm Sig to produce a response c_i. This response plays a role similar to that of a digital signature or a MAC depending on the situation. After producing c_i, the sender may execute a key evolution algorithm KeyEv that updates her secret key.

The receiver receives some subset of the message/response pairs $\{(m_i, c_i)\}$ of the sender in an arbitrary order, and must be able to determine whether each one is valid. The receiver does this by means of a verification algorithm that we denote by Ver.

On detecting a break-in attempt by the attacker in time step t, the sender initiates a key swapping algorithm denoted by Swap. This algorithm modifies

the internal state of the sender. The attacker may be assumed subsequently to gain access to all state information held by the sender, as well as all previous messages and responses.

The aim of an attacker is to try to forge valid responses on new, future messages $m_t, m_{t+1}, \ldots$ of his choice. The security of a funkspiel scheme has two aspects, which we elaborate on in section 4:

1. The attacker should be unable to determine whether or not the sender has indeed modified her internal state in response to the attack. We refer to this security property as *stealth*.

2. The attacker should be unable to create a response c_{t+u}, where $u \geq 0$, for any new message m_{t+u} such that the c_{t+u} is regarded as valid by the receiver. We refer to this security property as *unforgeability*.

These informal definitions are sufficient for an understanding of our proposed schemes. We offer more formal characterizations of the security of a funkspiel scheme in section 4.

3.1 Symmetric funkspiel scheme

The first scheme we describe is based on use of symmetric keys. In this scheme, the sender updates her secret key through application of a suitable pseudorandom generator f. Her sequence of secret keys is a sequence of successive images generated by f. The receiver, who shares the same initial secret key, is capable of producing an identical pseudorandom sequence without interaction with the sender. When she detects an attack, the sender substitutes a random key s'_i for her present secret key s_i. The security properties of f ensure that an attacker cannot derive significant information about s_{i-1} from s_i, and therefore cannot check whether s'_i is consistent with past history. Additional features of this solution include use of MACs as an authentication mechanism, as well as use of indices to synchronize the view of the receiver with that of the sender.

Let $f : \{0,1\}^j \rightarrow \{0,1\}^{j+k}$ denote a *pseudorandom generator* that stretches an arbitrary j-bit input to one of length $j+k$. In practice, f may be instantiated through, e.g., a conventional hash function such as SHA-1 [1]. We omit details here, instead referring the reader to [19] for a general treatment of the topic. (Of theoretical interest is the fact that f may be realized given the existence of any one-way function [16].)

Let $f_1(x)$ denote the first j bits of the output of f on input x, and let $f_2(x)$ denote the last k bits. Let $MAC : \{0,1\}^k \times \{0,1\}^* \rightarrow \{0,1\}^l$ be a *message authentication code* (*MAC*) that takes as input a k-bit secret key and a message of arbitrary length, and outputs an l-bit authentication code. (See, e.g., [20] for a discussion of MACs.) We use the symbol $\|$ to denote string concatenation. Variables j, k, and l are security parameters in this scheme. The algorithms are as follows:

Key setup (KeySet): The sender and receiver share the output of a key setup algorithm KeySet. The output of the algorithm is a secret key $s_1 = (\sigma_1, 1)$, where σ_1 is a j-bit string generated uniformly at random, and the second element of the pair is an index number.

Authentication code generation (Sig): The algorithm Sig takes as input a secret key $s_i = (\sigma_i, i)$ and a message $m_i \in \{0,1\}^*$. The output is a code $c_i = (MAC_{f_2(\sigma_i)}[m_i \parallel i], i)$. Given that the receiver may not receive messages in the order in which they are sent, concatenation of the index i is important here to ensure a binding between the message and the correct MAC key.

Key evolution (KeyEv): The algorithm KeyEv takes as input a secret key $s_i = (\sigma_i, i)$ and outputs $s_{i+1} = (\sigma_{i+1} = f_1(\sigma_i), i+1)$.

Key swapping (Swap): Input to the key swapping algorithm Swap is a secret key $s_i = (\sigma_i, i)$. Output is a secret key $s_i' = (\sigma_i', i)$, where σ_i' is a j-bit string generated uniformly at random. In other words, when the sender detects a tampering attempt, she simply overwrites her secret key with a random bit string.

Verification (Ver): The verification algorithm Ver takes as input a message/code pair (m, c'), where $c' = (a', i)$ for some integer i, and also the secret key s_1. The verification algorithm computes $a = MAC_{f_2(\sigma_i)}[m \parallel i]$. In other words, the verification algorithm checks the purported message authentication code on m using the index i. If $a = a'$, then the algorithm outputs '1'. Otherwise it outputs '0'.

Remarks. Note that if authentication codes are received in order, the verification algorithm may be made more efficient by having the receiver store s_i for use in the next verification. Even if authentication codes are received out of order, storage of seeds can still be used to reduce computational costs. Alternatively, storage costs may be reduced on the receiver side through derivation, rather than storage, of σ_1. In particular, the receiver may derive $\sigma_1 = h(ID_{sender}, S)$, where ID_{sender} is a unique identification number associated with the sender, while S is a master secret held by the receiver.[3]

More efficient use of f is possible for general pseudorandom generator constructions involving one-way permutations and hard-core bits. For example, it

[3]Of course, it is possible to eliminate use of message indices entirely. Rather than deriving the MAC for m_i through knowledge of i, the receiver can simply perform a search for the correct MAC key up to some predetermined number of steps, starting with σ_1. This can be costly, and slightly weakens the security of the scheme, but may be appropriate in some circumstances.

is possible to use the Blum-Blum-Shub generator [8], regarding the state of the generator as the output of f_1 and a sequence of hard-core bits as the output of f_2. This is more efficient than the scheme described above, in which it is implied that the output of f_1 itself consists of hard-core bits. The authors wish to thank an anonymous reviewer for this observation.

Finally, we note that in practice, the parameter l can be made quite small, e.g., on the order of 10 bits. Although this weakens the collision resistance property substantially, the funkspiel scheme will still work with well over 99% probability under straightforward assumptions on the underlying MAC. For the sake of simplicity, we omit investigation of the relevant security considerations from this paper.

3.2 Asymmetric funkspiel scheme

Our aim in proposing an asymmetric funkspiel scheme is twofold. First, we eliminate the use of indices associated with the authentication codes. Second, we achieve some strengthening of the security of the scheme. In contrast to our symmetric scheme, our asymmetric funkspiel scheme is resistant to forgery of authentication codes by an attacker with passive access to the private information of the receiver. (Note that our asymmetric scheme is not, however, resistant to detection of key swapping by such an attacker. The *stealth* property, as defined in section 4 is lost in this case, but the *unforgeability* property is retained.) As an additional benefit, we eliminate the need for the algorithm KeyEv and the associated key management and erasure. Barring a detected break-in by the attacker, the private information of the sender remains static throughout the lifetime of the scheme. In exchange for these benefits, our asymmetric scheme incurs greater overhead in terms of both communication and computational costs.

The idea behind the scheme is very simple. By sending authentication codes as ciphertexts under a semantically secure encryption algorithm (as defined in [15]), we prevent the attacker from seeing these codes. In consequence, on breaking into the device of the sender, the attacker cannot determine whether the private key contained therein is correct.

Let SigKeyGen be an algorithm that takes as input a security parameter $k \in Z^+$ and outputs a private/public signature key pair (SK, PK). Let $\Sigma_{SK}[m]$ denote a signature on message m under the corresponding signature algorithm with private key SK. Let $Ver_{PK}[\Sigma]$ be the verification operation on signature Σ using PK, with output '1' if the verification is successful, and '0' otherwise.

Similarly, let EncKeyGen be an algorithm that takes as input a security parameter $k \in Z^+$ and outputs a private/public encryption key pair (SK, PK). We assume that the encryption algorithm is semantically secure. We might, for example, use the El Gamal encryption algorithm [14, 24]. Generation of a ciphertext in such a cryptosystem requires a random encryption factor as input. Let $E_{PK,\rho}[m]$ denote an encryption of message m under public key PK using

random encryption factor ρ, and let D_{SK} denote the corresponding decryption operation.

In an initialization phase, the receiver produces an encryption key pair $(SK_r, PK_r) \leftarrow \mathsf{EncKeyGen}(\mathsf{k})$. The public key PK_r is subsequently certified and published. Our funkspiel scheme thus consists of the following algorithms.

Key setup : The receiver generates a signature key pair $(SK_s, PK_s) \leftarrow \mathsf{SigKeyGen}(\mathsf{k})$, stores PK_s, and sends (SK_s, PK_s) to the sender over a secure, authenticated channel. Alternatively, the sender generates the key pair and sends the PK_s to the receiver, likewise over a secure, authenticated channel.

Authentication code generation (Sig): The algorithm Sig takes as input the secret key SK_s, a message $m_i \in \{0,1\}^*$, and a random seed $\rho_i \in_u \{0,1\}^l$, where $\in_u$ denotes uniform random selection[4], Output is $c_i = E_{PK_r,\rho_i}[\Sigma_{SK_s}[m_i]]$.

Key swapping (Swap): Input to the key swapping algorithm Swap is the security parameter k. Output is a new key pair $(SK', PK') \leftarrow \mathsf{SigKeyGen}(\mathsf{k})$. In other words, when the sender detects a tampering attempt, she simply overwrites her private key with a new private key.

Verification (Ver): The verification algorithm Ver takes as input a message/code pair (m, c'), The output is simply $Ver_{PK_s}[D_{SK_r}[c']]$. In other words, the receiver decrypts the received code and then verifies the plaintext signature against the public key of the sender.

Remarks. The scheme as described relies wholly on expensive public-key operations. Of course, it is possible to construct any of a number of hybrid schemes involving use of both symmetric and asymmetric techniques. For example, rather than signing messages, the sender can compute a MAC using a shared key s. The use of semantically secure encryption in this case still eliminates the need for a key evolution algorithm, although in this hybrid scheme an attacker with access to the private information of the receiver can forge authentication codes. The lack of key evolution can open up other nonmathematical avenues of attack. In particular, a key swap might be detected from unusual power consumption or other abnormal electrical activity. Analysis of such attacks is beyond the scope of this paper, hence we confine ourselves to mention of the possibility.

[4]It is possible to generate ρ_i using a pseudorandom generator in a manner that is forward secure. In other words, it should be infeasible to determine previous outputs from the current output. This is achievable, for example, using the idea underlying the first funkspiel scheme, i.e., the pseudorandom generator should be used to produce both a randomization factor for the encryption and a new pseudorandom seed.

Another source of inefficiency is the need to re-derive a new key pair in the key swapping algorithm Swap. This problem can be addressed in most discrete-log based signature schemes, such as DSS [2], as follows. The secret key in such schemes consists of an integer $x \in Z_q$ for some 160-bit prime q. It suffices therefore for the key swapping algorithm to select a new integer $x' \in_u Z_q$. In fact, the key swapping algorithm can be made even more efficient if the sender simply flips a large set of low order bits – say, the first 100 bits – independently at random. (There is only a negligible chance that this will yield $x' \geq q$.) Provided that the sender does not store PK_s, an attacker will be unable to determine that x' is not the original private key of the sender.

3.3 Backward-malleable funkspiel scheme

Both of the funkspiel schemes proposed above rely on the sender producing a false secret key. This key is not in fact consistent with the transcripts produced by the sender. Instead, the security of these funkspiel schemes relies on the fact that the attacker is computationally incapable of determining whether the false secret is consistent with the previous outputs of the sender. An interesting problem is to find a funkspiel scheme in which the sender can demonstrate to the attacker that the false secret is consistent with previous outputs by the sender, but such that this false secret in fact yields invalid future outputs. We call this a *backward-malleable* funkspiel scheme. In contrast to the funkspiel schemes above, which offer cryptographic security, the backward-malleable funkspiel scheme we present here may be constructed so as to offer unconditional, i.e., information theoretic security. Backward-malleable funkspiel schemes are also of interest as an extension of the idea of duress PINs.

The aim in a backward-malleable funkspiel scheme is for the sender to store information such that when the attacker breaks in, and the key swapping algorithm Swap is executed, the sender device still contains private key material that is consistent with previous outputs. More precisely:

Definition 1 *A funkspiel scheme is* backward malleable *if there is an algorithm A_{check} polynomial in all security parameters such that the following holds for any polynomial time attacker A and any set of sender transcripts $(m_1, c_1), (m_2, c_2), \ldots, (m_j, c_j)$. Suppose that* Swap *is triggered on break-in by the attacker in time step j, yielding state s' in the sender device. The output $s^* = A_{check}(s')$ is a valid initial state for the sender device such that when reset to initial state s^*, the sender device yields authentication codes $c_1, c_2, \ldots, c_j$ on input messages $m_1, m_2, \ldots, m_j$.*

Note that there is no clear way to convert the funkspiel schemes described into backward malleable ones. For example, for the asymmetric scheme to be rendered backward malleable, the attacker would have to be given or be able to determine a tape R containing random encryption factors for ciphertexts $c_1, c_2, \ldots, c_j$ such that the decryption of these ciphertexts would be correct

plaintext signatures under signature key s' – even though these plaintext signatures were really produced using a different signature key.

The property of backward malleability here is essentially achievable using a deniable encryption scheme. Recall that the sender of a ciphertext c in such a scheme is able to produce two valid but distinct plaintexts m and m'. Thus, one way to achieve a backward-malleable funkspiel scheme is for the sender to substitute a secret s' for secret s and then, for previous deniable encryptions c_i, produce a valid plaintext corresponding to $MAC_{s'}[m_i]$ to convince the attacker of the validity of s'. Deniable encryption, however, incurs a rather high overhead in terms of both communication and computation, so that this scheme is not terribly practical.

A more practical strategy is the following. The sender and receiver share two secrets and a sequence of bits. The sender uses one of the two secrets, depending on the bit value designated by the current index. There are two ostensible drawbacks to this scheme. First, it involves larger asymptotic storage requirements than the schemes above. Second, it permits a small number of forgeries on the part of the attacker with non-negligible probability after break-in. In practice, however, neither of these drawbacks appears to be serious. Moreover, like the asymmetric scheme, this scheme carries the benefit of not requiring a KeyEv algorithm with the associated complexities of key management for the sender. The sender must maintain an index counter, while the receiver must keep track of which indices have been used by the sender, in order to avoid replay attacks.

The scheme comprises the following algorithms.

Key setup (KeySet): The sender and receiver share the output of a key setup algorithm KeySet. The algorithm takes as input a security parameter $k \in Z^+$ and a scheme lifetime T. The output of the algorithm is a pair of random, secret keys σ_0 and σ_1 and a sequence of randomly generated bits $b_1, b_2, \ldots, b_T$, along with T. The index i is initialized to 1.

Authentication code generation (Sig): The algorithm Sig takes as input the secret keys σ_0 and σ_1 and bits $b_1, b_2, \ldots, b_T$ as well as T and the current index i. If $i > T$, then the algorithm outputs "expiration". Otherwise, the output of the algorithm is $(MAC_{\sigma_{b_i}}[m_i \parallel i], i)$. The index i is then incremented.

Key swapping (Swap): Input to the key swapping algorithm Swap is the sequence of bits $b_1, b_2, \ldots, b_T$ and the current index i. The key swapping algorithm replaces b_t with $b'_t \in_U \{0, 1\}$ for all $t \in [i, i+1, \ldots, T]$ and outputs the new bit sequence $b_1, b_2, \ldots, b_i, b'_{i+1}, b'_{i+2}, \ldots, b'_T$. If $i > T$, then the key swapping algorithm outputs the input bit sequence with no changes.

Verification (Ver): The verification algorithm Ver takes as input a message/code pair (m_i, c'), where $c' = (a', i)$ for some integer i. Additional inputs are the secret keys σ_0 and σ_1 and the bit sequence $b_1, b_2, \ldots, b_T$. The algorithm first checks that the index i has not been used previously by the sender, and that $i \leq T$; if either condition is violated, then the algorithm outputs '0' and halts. Otherwise, the algorithm computes $a = MAC_{\sigma_{b_i}}[m_i \parallel i]$. If $a = a'$, then the algorithm outputs '1'. Otherwise it outputs '0'.

Remarks. This scheme may easily be seen to satisfy the definition of backward malleability. The algorithm A_{check} simply resets the counter i to 1. This backward-malleable funkspiel scheme may really be viewed as a kind of one-time pad underlying an efficient authentication algorithm. While the storage requirements of this scheme are linear in the scheme lifetime T, in practice they are small. Tradeoffs are possible between the storage efficiency of the scheme and the ability of the attacker to forge authentication codes successfully. As parameterized above, the probability of the attacker forging d valid codes after break-in is 2^{-d}. In an alternate scheme, the algorithm Sig uses σ_0 for t_1 steps, then σ_1 for t_2 steps, etc., where t_j may be fairly large. The key swapping algorithm randomly perturbs the lengths of these intervals. This results in a scheme with better storage efficiency, but in which the attacker may perform forgeries more successfully. Conversely, we could use ℓ bits for each message. If ℓ is reasonably small we could have 2^ℓ different secrets, or we might simply append ℓ bits to a universal secret σ input to the MAC.

3.4 Funkspiel deployment: Avoiding denial-of-service attacks

As in the ATM example presented in section 2.1, we may assume that on receipt of an invalid message/code pair (m, c), the receiver concludes that the sending device has been compromised. In this case, the receiver takes some kind of defensive action, perhaps seeking to trace the device or user of the device, or sounding an alarm. Thus, the funkspiel schemes as described above are vulnerable to a form of denial-of-service attack. An attacker capable of modifying the communications between the sender and receiver can simply corrupt a funkspiel code c, triggering a false alarm on the part of the receiver.

To address this problem in practice, it is necessary to employ an additional layer of message authentication on top of that provided by the funkspiel scheme. Often, a funkspiel scheme may simply be layered under an existing authentication mechanism. For example, in the ATM example in section 2.1, the funkspiel code was included as part of the signed message produced by the card. In general, the sender may use an independent secret s or private key SK as a means of providing message integrity on the pair (m, c) during transmission. The key s or SK may be protected from the attacker using conventional means. When

such an outer MAC or signature is applied, the inner MAC used in the described schemes can be dropped, and the associated keying material instead inserted explicitly. This is true since, as established below, the security properties against an attacker that obtains the information of the sender after the key swap do not depend on the existence of this MAC. The purpose of this inner MAC was simply to defend against less powerful attacks in which the private information of the sender is not compromised.

4 Security

There are two aspects to the security of a funkspiel scheme. First, an attacker breaking into the device of the sender should not be able to detect the use of the algorithm Swap, a property we refer to as *stealth*. Second, assuming successful deployment of Swap, an attacker should be unable to forge a correct authentication code even after breaking into the device. We refer to this latter property as *unforgeability*. It is easy to see that neither of these security characteristics implies the other. Below we give some properties achieved by our construction together with some sketches of proofs. For cryptographic definitions we appeal to the main body of cryptographic literature, including such standard texts as [19, 20]. In particular, we assume that all algorithms are efficient in the sense that they run in expected polynomial time; we refer to a probability as negligible if it is smaller than any inverse polynomial in the relevant security parameters.

4.1 Stealth security

The stealth property is best characterized in terms of the following experiment. The sender executes KeySet with the receiver, as in the normal protocol. The attacker then chooses a sequence of messages $m_1, m_2, \ldots, m_i$ in an adaptive manner, and is permitted to view the resulting authentication code sequence $c_1, c_2, \ldots, c_i$. For an i of its choice, polynomial in the security parameters, and again determined adaptively, the attacker subsequently declares that he will break into the device of the sender. At this point, the sender flips a coin. If the outcome is heads, she executes KeySwap; if the outcome is tails, she does nothing. The attacker now gains access to the full internal state of the device of the sender, and must determine whether KeySwap has been executed. If the stealth property holds, then the attacker will be able to do so with only negligible advantage.

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 authentication codes $(m_1, c_1), (m_2, c_2), \ldots, (m_i, c_i)$ 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 s_{i+1} and a message/code sequence $(m_1, c_1), (m_2, c_2), \ldots, (m_i, c_i)$. For $i = 0$, this may be regarded as a null sequence. The output of A_{guess} is '0' if the algorithm has determined that the sender executed Swap and '1' otherwise.

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

Experiment $STEALTH(\mathsf{FAC}, \mathsf{A})$
$s_1 \leftarrow \mathsf{KeySet}(k)$
$i \leftarrow 1$
$m_1 \leftarrow A_{\mathsf{Intr}}(\phi)$
while $m_i \neq$ "break" do
    $c_i \leftarrow \mathsf{Sig}(s_i, m_i)$
    $i \leftarrow i + 1$
    $s_i \leftarrow \mathsf{KeyEv}(s_{i-1})$
    $m_i \leftarrow A_{\mathsf{Intr}}((m_1, c_1), \ldots, (m_{i-1}, c_{i-1}))$
select $y \in_u \{0, 1\}$
if $(y = 1)$ then
    $\mathsf{s_i} \leftarrow \mathsf{Swap}(s_i)$
$g \leftarrow A_{\mathsf{guess}}(s_i, (m_1, c_1), \ldots, (m_{i-1}, c_{i-1}))$
if $g = y$ then
    return '1'
else
    return '0'

Definition 2 *Let* FAC *be a funkspiel authentication code 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.*

The stealth property of the symmetric funkspiel scheme may be seen to depend on the properties of the underlying pseudorandom generator. Let us prove that the scheme is secure in the case when the output of the generator cannot be efficiently distinguished from truly random bits. The key lemma is given below.

Lemma 1 *Suppose the pseudorandom generator f in the symmetric funkspiel has an output that is indistinguishable from a truly random bitstring. Then, for any $i \geq 0$ the sequence $(f_2(\sigma_\ell)_{\ell=1}^{i}, \sigma_{i+1})$ cannot be distinguished from a truly random bitstring of length $ik + j$.*

Proof: (Sketch) Fix the value of i and define hybrid distributions D_m, $1 \leq m \leq i + 1$. These are all closely related to the distributions given in the lemma and are defined as follows.

For $\ell < m$, replace the value $f_2(\sigma_\ell)$ for $\ell < m$ by a random bitstring, set σ_m to a random bitstring, and calculate σ_ℓ for $\ell > m$ as in the symmetric funkspiel scheme.

Clearly D_1 gives the same distribution as in the symmetric funkspiel scheme while D_{i+1} picks all strings uniformly at random.

If the conclusion of the lemma is false then D_1 and D_{i+1} can be efficiently distinguished and hence there is some m such that we can distinguish D_m and D_{m+1}. Now we claim that this implies an experiment such that we can distinguish the output of f from random bits.

In particular, given $j + k$ bits as input, we replace $f_2(\sigma_\ell)$ with a random bitstring for $\ell < m$. We then let the given $j+k$ bits compose $f_2(\sigma_m)$ and σ_{m+1}, and compute the rest of the output as above. If the given bits are random we have constructed an element from D_{m+1}, while if they were pseudorandom we have an element from D_m. The lemma follows. ∎

Clearly the above lemma implies stealth security for the symmetric funkspiel scheme. For the asymmetric scheme, stealth follows from the semantic security of the underlying encryption scheme.

Theorem 2 *If the public key encryption function used by the receiver in the asymmetric funkspiel scheme is semantically secure then that scheme has the stealth security property.*

Proof: (Sketch) Assume not. This implies that the attacker can distinguish between the two distributions

$$(E_{PK_r,\rho_\ell}[\Sigma_{SK_s}[m_\ell]])_{\ell=1}^{i}, SK_s$$

and

$$(E_{PK_r,\rho_\ell}[\Sigma_{SK_s}[m_\ell]])_{\ell=1}^{i}, SK'_s,$$

where SK_s and SK'_s are private keys chosen uniformly at random. We define hybrid distributions D_j by replacing the first j messages in the second distribution by ciphertexts of signatures under the key SK'_s. For $j = 0$ this yields no change, while for $j = i + 1$ we in fact have the distribution seen by the attacker. Thus if the stealth property is violated, it is possible to efficiently distinguish between D_0 and D_i and hence for some j we can distinguish between D_j and D_{j+1}.

This in its turn implies that we can efficiently generate two messages such that such that the attacker can distinguish between corresponding ciphertexts. It follows that the encryption scheme does not have the indistinguishability property and hence (by [15]) is not semantically secure. ∎

For the backward-malleable funkspiel scheme, stealth security is obvious since the distributions obtained before and after Swap are identical.

Theorem 3 *Stealth security holds for the backward-malleable funkspiel scheme.*

4.2 Unforgeability

We define unforgeability in terms of the ability of an attacker to forge authentication codes after break-in, and under the assumption, of course, that the Swap algorithm has been triggered. Having already established stealth security unforgeability comes naturally. We assume that we use schemes as they are given originally in sections 3.1, 3.2, and 3.3, without the outer authentication discussed in section 3.4. Once a definite outer authentication method based on sound principles has been decided on, similar results should follow in a straightforward manner.

We need to make a security assumption on our MAC. For simplicity we assume that it is *collision intractable*, i.e., that it is computationally infeasible to find two different strings that map to the same MAC. Weaker assumptions are sufficient to establish some properties. All that is needed is that ignorance of some particular input bits to the MAC implies similar ignorance of the output.

Theorem 4 *Suppose that the output of f is indistinguishable from a truly random bitstring and the MAC is collision intractable. Then the probability of a successful forgery in the symmetric funkspiel scheme can exceed 2^{-k} by only a negligible amount.*

Proof: (Sketch) By reasoning similar to that in the proof of Lemma 1, an attacker cannot distinguish $f_2(\sigma_i)$ from random bits for any i. The probability that the attacker can guess those bits correctly can hence exceed 2^{-k} only by a negligible amount. The theorem follows from the assumed property of the MAC. ∎

Next we turn to the asymmetric scheme. We need some security property of the signature scheme to establish security and almost any minimal requirement is sufficient. Let us assume a simple property which is non-standard. We say that a signature scheme is *diverse* if, for any message m the probability (over a random set of keys) that a given string s is a correct signature for m is negligible.

Theorem 5 *For the asymmetric funkspiel scheme with a semantically secure encryption scheme used by the receiver and a diverse signature scheme used by the sender, the probability that an attacker can violate unforgeability is negligible.*

Proof: (Sketch) As established in the proof of Theorem 2 the attacker cannot distinguish the correct signature key from a random signature key. Thus the probability that any produced c_i is correct is, by diversity, negligible. ∎

In the case when the secret key of the receiver is compromised, a good signature scheme still guarantees unforgeability. In this case, we essentially have an ordinary signature scheme. Since security definitions and properties in this case are standard, we do not state them here.

Finally we consider the backward-malleable funkspiel scheme.

Theorem 6 *Consider the backward-malleable funkspiel scheme used with a collision intractable MAC. The probability of a successful forgery exceeds 1/2 only by a negligible amount. The probability of detection is independent for each forgery attempt.*

Proof: (Sketch) Follows straightforwardly from the stealth security properties and the security property of the MAC. ∎

Finally note that for the variants of the backward-malleable funkspiel in which we use more or fewer bits in each message, it is straightforward to prove similar statements.

5 Conclusion: Further Directions

The most critical area for further exploration of funkspiel schemes is their application to secure hardware devices in current use. Smartcards, of course, are the most widely deployed tamper-resistant computing devices, and an important candidate platform for deployment of the idea. At present, however, secure processors on smartcards do not typically include contiguous power supplies, so that it is not feasible for them to perform computation when a break-in is detected. (Indeed, "zeroizing" cannot be deployed as a countermeasure on smartcards for this reason.) Contiguous power supplies do, however, form a part of secure modules for larger devices, such as the Dallas Semiconductor 1954 and IBM 4758 modules. It is possible, moreover, that contiguous power supplies will become a feature of smartcards in the future, given the relative effectiveness of "zeroization" in comparison with other, more passive countermeasures [18]. We may also expect the proliferation of handheld computing devices such as PalmPilots to provide platforms for more substantial security modules. Even given the availability of contiguous power supplies, a number of security engineering questions remain. Is it possible – or necessary, for that matter – to prevent power-analysis or timing attacks against funkspiel schemes? How much computation can such a tamper-resistant device be expected to perform in the course of an attack? Finally, there is the interesting question of whether a funkspiel scheme can be profitably deployed in software. Are there software agents or platforms on which a tampering attempt can be effectively detected? Online systems with intrusion detection mechanisms seem a particularly promising avenue of exploration in this regard. With use of a funkspiel scheme, it is potentially possible to identify and track the actions of an attacker after a break-in without having to resort to shutting down the system.

Deployment of funkspiel schemes also raises some different fraud management issues. Should a funkspiel scheme be deployed as a publicly announced security feature, or a concealed one? If the existence of a funkspiel countermeasure is made known, then it serves as a deterrent to fraud, as an attacker can never be certain that he will not be vulnerable to tracing. On the other hand,

an attacker aware of the existence of a funkspiel scheme will know to attempt to circumvent it. One of the advantages of a funkspiel scheme, of course, is that the attacker cannot determine – from a cryptographic perspective, at least – whether his attempt at circumvention has been successful. If the funkspiel mechanism is to be deployed in a concealed fashion, then "zeroization" might also be deployed as a decoy or as an alternative first-line countermeasure.

Acknowledgments

Thanks to Burt Kaliski, Michael Steiner, and also the anonymous reviewers of this paper for their comments and suggestions.

References

[1] FIPS 180-1. Secure hash standard. In *Federal Information Processing Standards Publication 180-1*. U.S. Department of Commerce/N.I.S.T., National Technical Information Service, 1995.

[2] FIPS 186. Digital signature standard. In *Federal Information Processing Standards Publication 186*. U.S. Department of Commerce/N.I.S.T., National Technical Information Service, 1994.

[3] R. Anderson and M. Kuhn. Tamper resistance – a cautionary note. In *The Second USENIX Workshop on Electronic Commerce*, pages 1–11, 1996.

[4] R. Anderson and M. Kuhn. Low cost attacks against tamper-resistant devices. In M. Lomas *et al.*, editor, *Security Protocols, 5th International Workshop*, pages 125–136. Springer-Verlag, 1997. LNCS no. 1361.

[5] Y. Aumann and M.O. Rabin. Authentication, enhanced security and error correcting codes (extended abstract). In H. Krawczyk, editor, *Advances in Cryptology – Crypto '98*, pages 299–303. Springer-Verlag, 1998. LNCS no. 1462.

[6] M. Bellare and S. Miner. A forward-secure digital signature scheme. In M. Wiener, editor, *Advances in Cryptology - Crypto '99*, pages 431–448. Springer-Verlag, 1999. LNCS no. 1666.

[7] J. Benaloh and D. Tuinstra. Receipt-free secret-ballot elections. In *26th ACM Symposium on Theory of Computing (STOC)*, pages 544–553. ACM Press, 1994.

[8] L. Blum, M. Blum, and M. Shub. A simple unpredictable pseudo-random number generator. *SIAM Journal on Computing*, 15(2):364–383, 1986.

[9] D. Boneh, R.A. Demillo, and R.J. Lipton. On the importance of checking cryptographic protocols for faults. In W. Fumy, editor, *Advances in Cryptology - Eurocrypt '97*, pages 37–51. Springer-Verlag, 1997. LNCS no. 1233.

[10] M. Burmester, Y. Desmedt, and J. Seberry. Equitable key escrow with limited time span (or, how to enforce time expiration cryptographically). In K. Ohta and D. Pei, editors, *Advances in Cryptology - Asiacrypt '98*, pages 380–391. Springer-Verlag, 1998. LNCS no. 1514.

[11] R. Canetti, C. Dwork, M. Naor, and R. Ostrovsky. Deniable encryption. In B.S. Kaliski, editor, *Advances in Cryptology - Crypto '97*, pages 90–104. Springer-Verlag, 1997. LNCS no. 1294.

[12] G. Davida, Y. Frankel, Y. Tsiounis, and M. Yung. Anonymity control in e-cash systems. In R. Hirschfeld, editor, *Financial Cryptography '97*, pages 1–16. Springer-Verlag, 1997. LNCS no. 1318.

[13] C. Dwork, M. Naor, and A. Sahai. Concurrent zero-knowledge. In *STOC '98*, pages 409–418. ACM Press, 1998.

[14] T. El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31:469–472, 1985.

[15] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28:270–299, 1984.

[16] J. Håstad, R. Impagliazzo, L. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28:1364–1396, 1999.

[17] D. Kahn. *The Codebreakers*. Macmillian Publishing Company, 1996.

[18] O. Kömmerling and M.G. Kuhn. Design principles for tamper-resistant smartcard processors. In *USENIX Workshop on Smartcard Technology (Smartcard '99)*, pages 9–20, 1999.

[19] M. Luby. *Pseudorandomness and Cryptographic Applications*. Princeton Univ. Press, 1996.

[20] A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[21] K. Sako and M. Hirt. Efficient receipt-free voting based on homomorphic encryption. In B. Preneel, editor, *Advances in Cryptology – EUROCRYPT '00*, pages 539–556. Springer-Verlag, 2000. LNCS no. 1807.

[22] K. Sako and J. Kilian. Receipt-free mix-type voting scheme - a practical solution to the implementation of a voting booth. In L.C. Guillou and J.-J. Quisquater, editors, *Advances in Cryptology – EUROCRYPT '95*, pages 393–403. Springer-Verlag, 1995. LNCS no. 921.

[23] G. Simmons. Subliminal channels; past and present. *European Transactions on Telecommunications*, 5(4):459–473, 1994.

[24] Y. Tsiounis and M. Yung. On the security of ElGamal-based encryption. In *1998 International Workshop on Practice and Theory in Public Key Cryptography (PKC '98)*, pages 117–134. Springer-Verlag, 1998. LNCS no. 1431.