A PROTOCOL IMPROVEMENT FOR HIGH-BANDWIDTH ENCRYPTION USING NON-ENCRYPTING SMART CARDS

Rüdiger Weis

Praktische Informatik IV

University of Mannheim, Germany

`rweis@pi4.informatik.uni-mannheim.de`

Abstract At the moment smart cards are the only practicable pretty secure place to store secret keys. But their physical properties limit the encryption bandwidth. Fortunately new cryptographic "Remotely keyed encryption" protocols support fast encryption on a slow smart card. For the scheme described here, even a smart card without a built-in encryption function would do the job, e.g., a signature card [LuWe99]. We also show a new interface-compatible security improvement to the RaMaRK scheme [Luck97], which needs only a light software modification on the host system. The improvement provides better cryptographic security and makes a known plaintext attack by [BFN98] infeasible. Furthermore, we discuss some new implementation aspects which provide a huge margin of security.

Keywords: Smart cards, Remotely Keyed Encryption, Cryptographic File Systems.

1. INTRODUCTION

Many security-relevant applications store secret keys on a tamper-resistant device, a *smart card*. Protecting the valuable keys is the card's main purpose. Even though in recent years some interesting cryptographic [Weis97] and many very dangerous hardware attacks [WKF97] have been developed, smart cards provide much higher security than other storage systems.

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: [10.1007/978-0-387-35575-7_19](https://doi.org/10.1007/978-0-387-35575-7_19)

Because of physical limitations, smart cards are typically slow. Fortunately new cryptographic protocols make fast encryption on a smart card supported system possible.

Smart cards are often designed to support authentication or digital signatures rather than encryption. In this paper, we concentrate on the RaMaRK protocol [Luck97] and show a new easy-to-implement security improvement.

The RaMaRK protocol does not require the smart card itself to support encryption – support for hash functions, as built into many signature cards, is sufficient. In a world with many restrictions on the import, export or usage of encryption tools and far fewer restrictions regarding authentication or signature tools, this can be an important property.

2. REMOTELY KEYED ENCRYPTION

A *remotely keyed encryption scheme* (RKES) distributes the computational burden for a block cipher with large blocks between two parties, a *host* and a *card*. We think of the host as being a computer under the risk of being taken over by an adversary, while the card can be a smart card, protecting the secret key. We do not consider attacks to break the tamper-resistance of the smart card itself. The host knows plaintext and ciphertext, but only the card is entrusted with the key.

An RKES consists of two protocols: the *encryption protocol* and the *decryption protocol*. Given a B -bit input, either to encrypt or to decrypt, such a protocol runs like this: The host sends a *challenge value* to the card, depending on the input, and the card replies with a *response value*, depending on both the challenge value and the key.

The notion of *remotely keyed encryption* is due to Blaze [Blaz96]. Lucks [Luck97] pointed out some weaknesses of Blaze's scheme and gave formal requirements for the security of RKESs:

- (i) **Forgery security:** If the adversary has controlled the host for $q - 1$ interactions, she cannot produce q plaintext/ciphertext pairs.
- (ii) **Inversion security:** An adversary with (legitimate) access to encryption must not be able to decrypt and vice versa.
- (iii) **Pseudorandomness:** The encryption function should behave pseudorandomly for someone without access to the card, nor knowledge of the secret key.

While requirements (i) and (ii) restrict the abilities of an adversary with access to the smart card, requirement (iii) is only valid for *outsider adversaries* without access to the card. If an adversary could compute

forgeries or run inversion attacks, she could easily distinguish the encryption function from a random one.

3. THE RAMARK ENCRYPTION SCHEME

In this section, we describe the *Random Mapping based Remotely Keyed (RaMaRK) Encryption scheme*, which uses several independent instances of a *fixed size random mapping* $f : \{0, 1\}^b \rightarrow \{0, 1\}^b$. The scheme is provably secure if its building blocks are, i.e., it satisfies the requirements (i)–(iii) above, see [Luck97]. Note that b must be large enough to make performing close to $2^{b/2}$ encryptions infeasible. We recommend to choose $b \geq 160$. By ‘ $\oplus$ ’ we denote the bit-wise XOR, though mathematically any group operation would do the job as well.

We use three building blocks:

1. Key-dependent (pseudo-)random mappings

$$f_i : \{0, 1\}^b \rightarrow \{0, 1\}^b.$$

2. A hash function

$$H : \{0, 1\}^* \rightarrow \{0, 1\}^b.$$

H has to be *collision resistant*.

3. A pseudorandom bit generator (i.e. a ‘stream cipher’)

$$S : \{0, 1\}^b \rightarrow \{0, 1\}^*.$$

If the seed $s \in \{0, 1\}^b$ is randomly chosen, the bits produced by $S(s)$ have to be indistinguishable from randomly generated bits.

In addition to pseudorandomness, the following property is needed: If s is secret and attackers choose $t_1, t_2, \dots \in \{0, 1\}^b$ with $t_i \neq t_j$ for $i \neq j$ and receive outputs $S(s \oplus t_1), S(s \oplus t_2), \dots$, it has to be infeasible for the attackers to distinguish these outputs from independently generated random bit strings of the same size. Hence, such a construction behaves like a random mapping $\{0, 1\}^b \rightarrow \{0, 1\}^{B-2b}$, though it is actually a pseudorandom one, depending on the secret s .

Based on these building blocks, we realize a remotely keyed encryption scheme to encrypt blocks of any size $B \geq 3b$, see the following figure.

We represent the plaintext by (P, Q, R) and the ciphertext by (A, B, C) , where

$$(P, Q, R), (A, B, C) \in \{0, 1\}^b \times \{0, 1\}^b \times \{0, 1\}^{B-2b}.$$

Figure 1.1 The RaMaRK encryption protocol

For the protocol description we also consider intermediate values

$$T, U, V, X, Y, Z \in \{0, 1\}^b, \text{ and } I \in \{0, 1\}^{B-2b}.$$

The encryption protocol works as follows:

1. Given the plaintext (P, Q, R) , the host sends P and Q to the card.
2. The card computes $U = f_1(P) \oplus Q$ and $T = f_2(U) \oplus P$, and sends $X = f_3(T) \oplus U$ to the host.
3. The host computes $I = S(X) \oplus R$ and $Y = H(I)$, sends $Z = X \oplus Y$ to the card, and computes $C = S(Z) \oplus I$.
4. The card computes $V = f_4(T) \oplus Z$, and sends the two values $A = f_5(V) \oplus T$ and $B = f_6(A) \oplus V$ to the host.

Decryption is done similarly. The description of decryption is omitted here to save space.

If the block size B of the cipher it realizes is not too small compared to the parameter b , the RaMaRK scheme is efficient. The card itself operates on $2 \cdot b$ bit data blocks, and both $3 \cdot b$ bit of information enter and leave the card.

4. EXTENDED SECURITY MODEL

Blaze, Feigenbaum (AT&T) and Naor (Weizmann Institut) [BFN98] published recently a paper on the EUROCRYPT'98 which has showed a new formal model for RKES, found a problem in the RaMaRK protocol and suggested a new RKES, that fulfills the new security model.

4.1 BFN MODEL OF PSEUDORANDOMNESS OF A RKES

It is theoretically desirable that a cryptographic primitive always appears to behave randomly to everyone without access to the key. In any RKES, the amount of communication between host and card should be less than the input length, otherwise the card could just do the complete encryption on its own. Since (at least) a part of the input is not handled by the smart card, and, for the same reasons, (at least) a part of the output is generated by the host, an insider adversary can easily decide that the output generated by herself is not random.

Blaze, Feigenbaum, and Naor [BFN98] found another way to define the pseudorandomness of RKESs. Their formal definition is quite complicated. It is based on the following scenario:

Adversary A gains direct access to the card *for a certain amount of time* and makes a fixed number of interactions with the card. Once A has lost direct access to the card, the encryption function should appear to behave randomly, even to A .

4.2 PROBLEMS OF THE RAMARK SCHEME

Regarding the RaMaRK scheme they pointed out that an adversary A who has had access to the card and lost the access again, can later choose special plaintexts where A can predict *a part of* the ciphertext. This makes it easy for A to distinguish between RaMaRK encryption and encrypting randomly.

The intermediate value X depends only on the (P, Q) -part of the plaintext, and the encryption of the R -part depends only on X . If A chooses a plaintext (P, Q, R) , having participated before in the encryption of (P, Q, R') , with $R \neq R'$, the adversary A can predict the C -part of the ciphertext, but not the P nor the Q part, corresponding to (P, Q, R) on her own.

Thus, according to the definition of [BFN98], the RaMaRK scheme is not pseudorandom.

4.3 DECRYPTION OF CIPHERTEXTS

Further the authors of [BFN98] pointed that there is a possibility to attack files with the same $2b$ bit header.

"However, because the encryption key depends only on the first two plaintext blocks, an arbitrarily large set of messages all of which start with the same two blocks will always be encrypted with the same key. This is not a hypothetical situation: A set of files in a computer file system, for example, might always start with the same few bytes of structural information.

The above describes a *known plaintext* distinguishing attack, that is actually feasible. The authors of [BFN98] continue:

An adversary that controls the host during the encryption or decryption of *one* file in such a set could subsequently decrypt the encryption of *any* file in the set."

We argue that this attack is *not* feasible. Note that the second intermediate key Z (resp. X for decryption) depends on all bit of the plaintext (P, Q, R) (resp. ciphertext (A, B, C)).

$$Z = X \oplus Y = X \oplus H(I) = X \oplus H(\boxed{R} \oplus S(X))$$

Thus the knowledge of the intermediate value X (resp. Z) is not sufficient for a decryption of *any* file of the mentioned set of files.

On the other hand it is a not satisfactory cryptographic property that an attacker can peel off one of the two stream cipher encryptions if she knows the intermediate key X .

$$C = I \oplus S(Z) = R \oplus S(X) \oplus S(Z)$$

5. A SIMPLE PROTOCOL IMPROVEMENT

Since there are several advantages of the RaMaRK scheme, the author suggests a slight modification of the protocol on the host side.

5.1 PROTOCOL MODIFICATION

We want to make sure that also the intermediate keys X and Y depend on every plaintext bit. Instead of P and Q we submit

$$\bar{P} := P \oplus h(R) \text{ and } \bar{A} := A \oplus h(C)$$

where h is a cryptographic hash function.

The *Improved RaMaRK scheme* is interface-compatible with the unmodified RaMaRK scheme. So no hardware modifications to the smart card are necessary.

5.2 CHARACTERISTICS AND LIMITATIONS

If we choose a standard hash function with 160-bit output, a *known plaintext attack* against the pseudorandom property seems to be infeasible.

According to Lucks, a *chosen plaintext attack* in the BFN scenario to distinguish the output of the protocol from a random output is still feasible. So even the improved RaMaRK scheme does not meet the stronger security model of [BFN98].

Further more it is not possible to peel off one stream cipher encryption as discussed in the last section.

The modification requires two expensive hash function calls for the big block B . We do not expect this to cause a problem for most applications since the main bottleneck seems to be the communication with the card. But if our improved scheme leads to performance problems on the host, we suggest using a fast, non-cryptographic hash function.

5.3 REMARKS ON THE BFN AND ARK SCHEMES

Lucks has recently published a new Accelerated Remotely Keyed Encryption (ARK) scheme [Luck99] which fulfills the BFN security model. The ARK scheme offers better performance and non-asymptotic security proofs.

Note that all schemes in [BFN98] and [Luck99] are pseudorandom as defined there, but depend on *pseudorandom permutations* (i.e., block ciphers) – and thus are designed for smart cards with built-in encryption.

6. IMPLEMENTATION OF BUILDING BLOCKS ON THE HOST

Hash Functions. In order to combine the big block of data with the small blocks in the card we need a collision-free hash function. The calculation is performed on the host, so we can simply choose a well-tested hash function like SHA-1 or RIPE-MD160. Both produce a 160-bit output, which seems to provide sufficient security.

Pseudo Random Bit generators. In [Luck97] the use of a stream cipher was suggested. But we can also use a well-tested block cipher in the OFB or CFB mode (E.g. CAST-5 performs very fine even on small packets [WeLu98]).

7. HASH BASED PRM

In this section we discuss how to realize Pseudo Random Mappings (PRM) with a Non-Encrypting smart card. One promising idea for realizing PRM on a smart card is to use a hash-based MAC function.

HMAC [BCK96] uses an iterative cryptographic hash function $\mathcal{H}$ as black box. Let ℓ be the bit length of the input of the compression function, then we define

$$\text{HMAC}_K(x) = \mathcal{H}(\bar{K} \oplus \text{opad} || \mathcal{H}(\bar{K} \oplus \text{ipad} || x))$$

with $\text{ipad} := \text{Ox36}$ repeated $\ell/8$ times and $\text{opad} := \text{Ox5C}$ repeated $\ell/8$ times, $\bar{K}$ is generated by appending zeros to the end of K to create a ℓ -bit string. (RIPEMD-160, SHA-1 and MD5 all use $\ell = 512$.)

This approach has several advantages. Cryptographic hash functions have been well studied; they are usually faster than encryption algorithms. It is easier in many countries, to export or import an authentication tool, such as a signature system, than to export or import an encrypting system.

Security of HMAC

In [BCK96] it was proven that HMAC provides security against collision and forging attacks with only weak assumptions on the underlying hash function.

This leaves an additional margin of security: even if some weakness in the hash function H (e.g. MD5) is found, the MAC based on H may still be secure. For example, a collision of hash function means finding a collision with a fixed Initial Vector (IV) and known output. If an attacker wants to find collisions in HMAC, he must find a collision in the underlying hash function even when the IV is random and secret, and the hash value is not explicitly known.

HMAC based on SHA-1 or RIPE-MD160 provides a 160-bit output. So even birthday attacks which need 2^{80} operations seem to be infeasible.

Note that a MAC based on a standard 64-bit block cipher (e.g. Triple-DES) is insufficient. Even MAC based on AES, which will be a 128-bit block cipher, cannot be recommended for high-security applications.

8. IMPLEMENTATIONS FOR SECURE DISTRIBUTED SYSTEMS

We are working on various hardware implementations of a Secure SmartCard Supported Filesystem ($\mathcal{S}^3\mathcal{F}$) and security solutions for various distributed multimedia applications developed at the Lehrstuhl Praktische Informatik IV of Prof. Dr. Effelsberg such as the digital lecture board [GeWe98] or the Education Multimedia Library [LWH99].

Acknowledgments

The author thanks Stefan Lucks for his critical remarks on the security of our protocol modification and implementation aspects and Wolfgang Effelsberg for many helpful comments on building the final version of this paper.

References

- [BCK96] Bellare, M., Canetti, R., Krawczyk, H., "Keying hash functions for message authentication", *Advances in Cryptology - Crypto 96 Proceedings*, Springer, 1996.
- [Blaz96] Blaze, M., "High-Bandwidth Encryption with Low-Bandwidth Smartcards", *Fast Software Encryption* (ed. D. Gollmann), Springer LNCS 1039, 33–40, 1996.
- [BFN98] Blaze, M., Feigenbaum, J., Naor, M., "A Formal Treatment of Remotely Keyed Encryption (Extended Abstract)", *Euro-*

- crypt '98, Springer LNCS 1403, 1998, pp. 251-265.
- [DBP96] Dobbertin, H., Bosselaers, A., Preneel, B., "RIPEMD-160, a strengthened version of RIPEMD", Proc. of Fast Software Encryption (ed. D. Gollmann), LNCS 1039, Springer, 1996.
 - [FIPS180] NIST, "Secure Hash Standard", Washington D.C., April 1995.
 - [GeWe98] Geyer, W., Weis, R., "A Secure, Accountable, and Collaborative Whiteboard", Proc. of IDMS '98, Oslo, Springer LNCS, 1998.
 - [Luck97] Lucks, S., "On the Security of Remotely Keyed Encryption", Fast Software Encryption, (ed. E. Biham) Springer LNCS, 1997.
 - [Luck99] Lucks, S., "Accelerated Remotely Keyed Encryption", Fast Software Encryption, Springer LNCS, Rome, 1999.
 - [LuWe99] Lucks, S., Weis, R., "Remotely Keyed Encryption Using Non-Encrypting Smart Cards". USENIX Workshop on Smartcard Technology, Chicago, May 10-11, 1999.
 - [LWH99] Lucks, S., Weis, R., Hilt, V., "Fast Encryption for Set-Top Technologies", Multimedia Computing and Networking '99, San Jose, 1999.
 - [Weis97] Weis, R., "Combined Cryptanalytic Attacks against Signature and Encryption schemes" (in German), A la Card aktuell 23/97, S.279, 1997.
 - [Weis98a] Weis, R., "Modern Block Ciphers" (in German), in: "Kryptographie", Weka-Fachzeitschriften-Verlag, Poing, 1998.
 - [Weis98b] Weis, R., "Modern Stream Cipher" (in German), in: "Kryptographie", Weka-Fachzeitschriften-Verlag, Poing, 1998.
 - [Weis98c] Weis, R., "Cryptographic One-Way-Functions" (in German), in: "Kryptographie", Weka-Fachzeitschriften-Verlag, Poing, 1998.
 - [WKF97] Weis, R., Kuhn, M., Floricic, B., "Hacking Chipcards", Workshop CCC '97, Hamburg 1997.
 - [WeLu98] Weis, R., Lucks, S., "The Performance of Modern Block Ciphers in JAVA", Preproceedings Cardis'98, Louvain-la-Neuve, Proceedings (ed Quisquarter, Schneier) will appear in Springer LNCS, 1998.