

How to Develop Export-Oriented Smart Solutions?

[Published in *Proc. of Gemplus Developer Conference*, Paris, France, June 21–22, 1999.]

Christophe Bidan and Pierre Girard

Gemplus, Cryptography & Security Group
B.P. 100, 13881 Gémenos Cedex, France
{Christophe.Bidan, Pierre.Girard}@gemplus.com

Abstract. Smart card based products are subject to the cryptographic export regulations. Unfortunately, these regulations are often taken into account too late, and it results in export restrictions. We claim that most of the times, these restrictions could be avoided by considering the cryptographic export matters at the earliest design stages. The aim of this paper is to provide a few principles to use cryptographic primitives. These principles will help the development of smart card based products that are consistent with the most common cryptographic regulations.

1 Introduction

Given the expansion of the Internet technology as the e-commerce, as well as the globalization of the data exchanges, the security (in the meaning of confidentiality and integrity) of the manipulated and/or sent data is a significant problem of today's computer field. In this context, the smart cards can play an essential role, as an useful and secure device for the mechanisms (e.g., the cryptographic functions and secret keys) allowing to guarantee the security. However, the products based on smart card technology must face the constraints of the export of cryptographic mechanisms.

Troubles related to cryptographic export regulations and laws often arise very late in the life cycle of a product, generally when the product is available and ready to ship. Most of the time, it results in export restrictions, or, at least, increases drastically the time to market of the product.

We claim that most of these problems could have been solved easily by taking into account the cryptographic export requirements at the earliest design stages. This is because most of the products don't need to offer cryptographic facilities to the end-user, which is subject to export regulations, but simply need to use cryptographic primitives in order to implement the security requirements.

The aim of this paper is to provide system designers, Value Added Resellers or whoever's concerned by a worldwide product deployment with a few tips to use cryptographic primitives, and avoid the common cryptographic regulations pitfalls. But first, let us recall the current legal framework.

2 The Legal Framework

We give an overview of the constraints related to the export of cryptographic functions, as defined in the Wassenaar arrangement, and we state the results of these constraints in the smart card field.

2.1 The Wassenaar arrangement and local laws

The Wassenaar arrangement is a multilateral arrangement on export controls for conventional weapons and sensitive dual-use goods and technologies. The 33 participating states, including Australia, most of the European Community, Japan, the Republic of Korea, New Zealand, Norway, the Russian Federation and the USA, maintain effective export controls on a agreed lists of items and exchange biannual notifications of goods transfers.

The export controls are implemented in each country with various degrees of strictness and effectiveness. The variability of local laws may be confusing, but one might beware of potential changes of most flexible legislation to comply with the arrangement. The countries actually implementing the arrangement literally are the USA and France.

2.2 What is subject to export regulations

A specific list of the arrangement (category 5, part 2) concerns the information security products. Basically, the products are controlled if providing cryptographic functions other than authentication or digital signature. It means that if a device can be used in a way or another as a ciphering/deciphering machine, it may be export-controlled.

One should notice that “weak” cryptography is not restricted, where “weak” cryptography means:

- A “symmetric algorithm” employing a key length up to 56 bits (e.g., DES [8]); or
- An “asymmetric algorithm” where the security of the algorithm is based on the factorisation of integers up to 512 bits (e.g., RSA [16]); or
- An “asymmetric algorithm” where the security of the algorithm is based on the computation of discrete logarithms in a multiplicative group of a finite field of size smaller than 521 bits (e.g., Diffie-Hellman [3]); or
- An “asymmetric algorithm” where the security of the algorithm is based on the computation of discrete logarithms in a group other than the previous one, and up to 112 bits (e.g., Diffie-Hellman over an elliptic curve [14]).

However, due to the growing performance of computers and/or cryptanalysis advances, the use of “weak” cryptographic functions may be under consideration only when data security (in term of confidentiality and/or integrity) is not a crucial property.¹ In other words, from the security point of view, products based on “weak” cryptographic functions should be considered as unsecured.

¹ For instance, it takes less than 9 days of electronic computation to break the DES algorithm [4].

We will not detail the cryptographic export policies any further. The reader can refer to Koops[13], who gives a survey of existing and proposed laws and regulations on cryptography, with entries per country on import/export controls, domestic laws, developments to restrict cryptography, and developments favoring cryptographic use. We can also mention Anderson [2], who discusses the legislation in Europe, and gives an overview of the currently fielded cryptographic applications. His conclusion is that most of applications use cryptography to prevent fraud (e.g., authentication), but a few of them is concerned with confidentiality. In other words, only few cryptographic products should be concerned by cryptographic regulations.

2.3 What is the implication in the smart card trade

In the context of the Wassenaar arrangement, smart cards are considered as (potential) ciphering/deciphering machine, and then, are under cryptographic export regulations. One should notice that a smart card that only implements digital signature functions, or “weak” cryptographic functions, does not fall into cryptographic export regulations. However, the main problem results in guaranteeing that the smart card behavior can not be modified so as to use it as a ciphering/deciphering machine, or to implement “strong” cryptographic functions.²

The aim of this paper is to provide principles for using cryptographic functions that help the development of smart card based products which could be exported according to the Wassenaar arrangement.

3 Design Principles

In this section, design principles will be illustrated with some examples. Let us precise that these principles are neither necessary, nor sufficient, and the final decision depends on each country laws. However, these principles are a first-step towards export authorization.

3.1 Basics

The design principles defined in this subsection are related to the usual mistakes found in the smart card based products. These mistakes generally result of a misunderstanding of both the security properties (e.g., confidentiality *vs.* integrity) and the cryptographic mechanisms used to guarantee these properties.

² For instance, DES and DES^{-1} allows to build 3DES, that is a “strong” cryptographic function.

Principle 1 *Do not use encryption unless it is really necessary.*

Encryption is not a security panacea and should not be used for other purposes than confidentiality. To avoid the unsuitable use of encryption, we advise to specify the security requirements in terms of confidentiality and integrity, from the product design stage. If you don't need confidentiality as a functionality but just integrity, secret-sharing or identity proof, try to achieve those purposes by others means (e.g., a keyed-MAC function such as keyed-MD5 or HMAC [11, 12]).

As an illustration, let us consider a smart card based product for which the smart card is used to secure the exchanges (*via* a computer network) between the end-user of the card and a remote server.³ We suppose that the confidentiality of the manipulated data is not required. In this context, the security of the exchanges consists in guaranteeing the data integrity, and the smart card does not need to implement encryption mechanisms. One solution may be to use a hash function (e.g., SHA-1 [7]) and a secret key K shared by the smart card and the server to build a keyed-MAC function. For instance, let M be the message to be signed, we define the digital signature as:

$$S = \text{SHA-1}(K | \text{SHA-1}(K | M)).$$

where $A|B$ denote the concatenation of A and B . Then, the sent message may be $M|S$.

This previous solution allows to guarantee the integrity of the exchanged data, any modification being detected.⁴ It does not use encryption functions, and then does not fall into export regulations.

Principle 2 *Do not leave the smart card working outside its application context.*

The use of a smart card outside its application context may allow, for instance, to play a given sensitive command an infinite time (i.e., commands which use encryption/decryption functions), and result in the unrestricted use of the card as a ciphering machine. This can occur since the use of cryptographic functions to perform authentication or digital signature is free regarding export regulations, but may be bypassed to achieve “strong” cryptographic functions.

In order to prevent the use of a smart card outside its application context, it is crucial to verify the execution order of sensitive commands as those used in the mutual authentication protocols, and to check the format of the APDU messages, as to guarantee that each message is the one expected.

³ In this example, we do not consider the authentication phase. We address it when we illustrate the principle 2.

⁴ However, notice that we do not consider replay attacks, otherwise the digital signature computation should be dependent on random and/or counter.

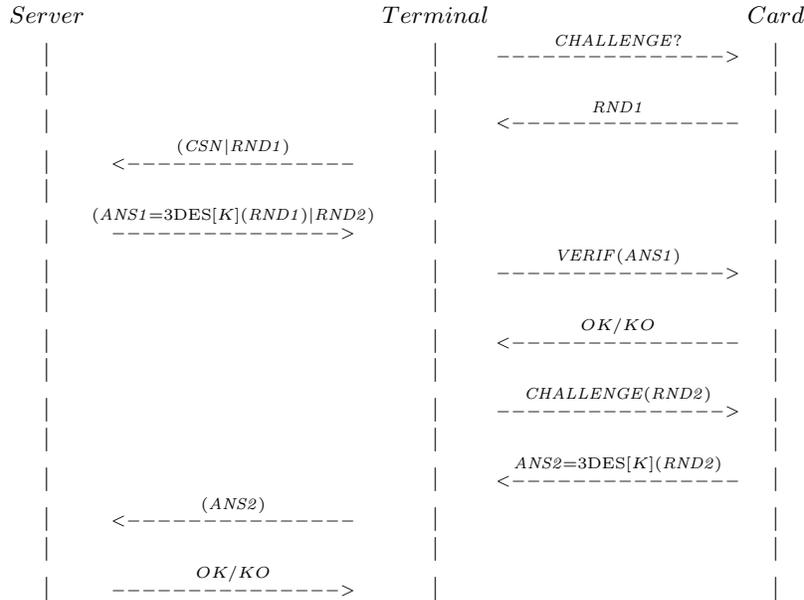


Fig. 1. Mutual authentication protocol with triple DES.

As an illustration, let us focus on the mutual authentication protocol of the previous example (figure 1).⁵ The terminal of the end-user asks a random $RND1$ to the card, and sends $(CSN|RND1)$ to the remote server, where CSN is the card serial number. Thanks to CSN , the remote server can retrieve the secret key K of the card, and sends back to the terminal $(ANS1|RND2)$, where $ANS1 = 3DES[K](RND1)$ denotes the triple DES encryption of $RND1$ using the key K , and $RND2$ is a random. The card allows to check that $RND1 = 3DES^{-1}[K](ANS1)$, and to compute $ANS2 = 3DES[K](RND2)$ that is sent to the server. Finally, the server checks that $RND2 = 3DES^{-1}[K](ANS2)$.

Let us now suppose that there is no constraints concerning the call of the function that allows to compute $ANS2$. Then, according to the Wassenaar arrangement, the protocol falls into export regulations, since we can use the previous function to encrypt any message. However, let us notice that the smart card can not be used as a decipher machine.

A simple solution to limit the use of the card as an encryption machine may be to only permit internal authentication (i.e., the call of the function that compute $ANS2$) after external authentication (i.e., checking $ANS1$). Another solution is to change the protocol so as to be compliant with the Wassenaar arrangement (figure 2). Briefly, the remote server sends back to the terminal $(ANS1)$, where $ANS1 = 3DES(RND1|RND2)$. Then, the card allows to check $ANS1$, and to

⁵ Let us notice that the protocol is not secure since we can perform some active attacks. Nevertheless, this protocol is sufficient to illustrate the previous principle.

compute $ANS2 = 3DES(RND2|RND1)$, that is sent by the terminal to the server.⁶

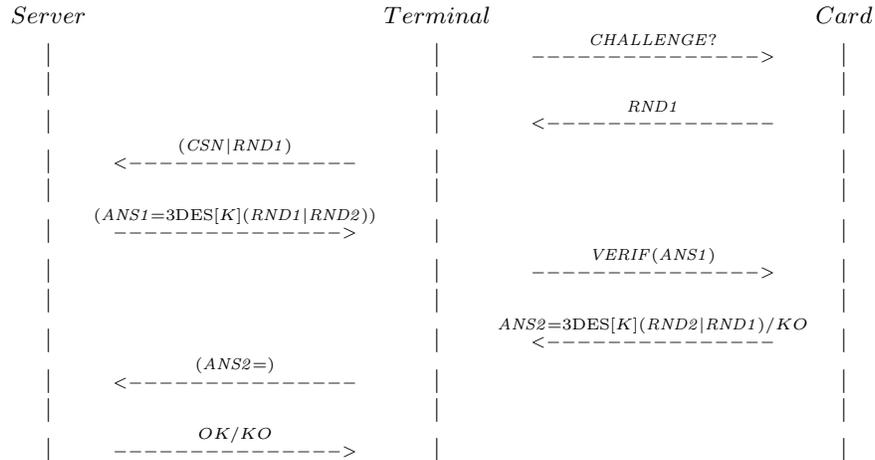


Fig. 2. Exportable mutual authentication protocol with triple DES.

3.2 Signing and MACing

As shown from experience, a number of cryptographic export problems are due to the possibility to hijack the digital signature functions so as to build cipher/decipher functions. The following design principles help to avoid this pitfall.

Principle 3 *Always hash a message you have to sign.*

Using a hash function before signing a message allows to ensure that the digital signature algorithm can not be used as an encryption algorithm. This principle is very useful because the export controls on encryption are stricter than those on digital signature. Then, using hash function permits to export products with “strong” digital signature functions but “weak” encryption functions.

We illustrate the previous principle with the use of the RSA algorithm. One characteristic of the RSA algorithm is that it can be used both as a digital signature scheme and an encryption scheme. Nevertheless, using a hash function

⁶ According to the principle 1, it should be better to use a hash based keyed-MAC function (e.g., HMAC-SHA) instead of a “strong” cryptographic function. Nevertheless, we have chosen to illustrate each principle independently.

(e.g., MD5 [10] or SHA-1) before signing ensures that the RSA digital signature algorithm implemented into smart cards can not be hijacked to encryption machines.

Let us point out that, for performance reasons, the hash function is generally performed outside the card. In particular, when the digital signature function is a keyed-MAC function, only the last block computation is performed on the card.

When the digital signature is not a keyed-MAC function, the smart card is only in charge of signing the hash value. But, in this case, it is easy to use the smart card as an encryption machine. The following principle addresses this problem.

Principle 4 *Use a padding whenever possible.*

The padding is used to characterize the digital signature functions from encryption functions, for the outside of the card. In that way, we ensure that the digital signature functions can be free of access without allowing to build decryption functions.

In particular, when the hash function is computed outside the card (i.e., by the terminal), the digital signature function has to be performed with a padding. First, it allows to limit the cipher power of the card, and also, it ensures that the card can not be used as a decipher machine.

As an illustration, suppose that *Sign* is the digital signature function of the card. Given a message M , the card computes the digital signature $S = \text{Sign}(M|P)$ where P denotes the padding. Even if the card is used as a cipher machine, it can not be used as a decipher machine since, given an encrypted message E , $\text{Sign}(E|P)$ does not allow to decipher E .

Principle 5 *Do not use a block cipher algorithm as a digital signature algorithm.*

Any block cipher algorithm may be used to build a hash function, and then a keyed-MAC function [15]. For instance, the 3DES in CBC mode [9] allows to implement a digital signature algorithm (also called 3DES-MAC [6]): the digital signature is done by the last eight-bytes result. This algorithm is often used in smart card, and even recommended by some standards (e.g., EMV [5]). However, when the data to be signed is eight-bytes long, the digital signature corresponds to the encryption of the data using 3DES. As a consequence, any smart card using the 3DES in CBC mode as a digital signature algorithm is subject to export restrictions.

We illustrate the use of the 3DES-MAC as a cipher/decipher algorithm. First, we recall the use of the 3DES in CBC mode. Let $M = (M_1|M_2|\dots|M_n)$ be a message where M_i , for $i = 1$ to n is eight-bytes long. Let C_i , for $i = 1$ to n , be the cipher text defined by:

$$C_i = 3DES[K](M_i \text{ xor } C_{i-1})$$

where K is the key used to encrypt, and $C_0 = 0$. Then, the encryption of M using the 3DES in mode CBC is $C = (C_1|C_2|\dots|C_n)$, and the digital signature of M using the 3DES-MAC is defined by C_n .

If $M = M_1$ (i.e., the message to be signed is eight-bytes long) then, according to the 3DES-MAC algorithm, the digital signature of M is $C_1 = 3DES[K](M_1 \text{ xor } C_0) = 3DES[K](M_1)$, that is, the encryption of M using the triple DES algorithm.

One solution to avoid the export restriction resulting of using the 3DES in CBC mode could consist in sending only the four least/most significant bytes of the 3DES-MAC result. However, it is then trivial to find a collision, that is two different messages with the same signature. In other words, the use of the four least/most significant bytes of the 3DES-MAC result should be considered as a “weak” digital signature function.

Finally, instead of using the 3DES in mode CBC as a digital signature algorithm, we recommend to use a hash function to build a keyed-MAC function.

Principle 6 *Do not use the same key for both signature and encryption*

According to the Wassenaar arrangement, digital signature may be based on “strong” cryptographic functions without export restrictions, whereas the use of “strong” cryptographic function to encrypt data is subject to export control. When a product based on smart card is not subject to export constraints, and requires “strong” encryption algorithm (i.e., if the main goal of the product is data confidentiality), the smart card can implement “strong” algorithms for both encryption and digital signatures. Otherwise, using two different keys for signature and encryption allows to implement a “strong” digital signature algorithm but a “weak” encryption algorithm.

Finally, one should notice that using the same key for both signature and encryption may result in allowing attacker to fake certificates or digital signature. The use of the same key also brings about using the key twice more, which is less secure than have different keys for each operation.

As an illustration, let us consider a smart card based product for which the card is used to secure the exchanges between the end-user and his/her bank. Such a product needs an authentication protocol to perform the mutual authentication between the end-user and the bank server, a digital signature algorithm to ensure the integrity of the exchanged data, and an encryption algorithm to provide privacy of the exchanged data.

If the product uses “strong” cryptography functions for both authentication, signature and encryption, it falls systematically into cryptographic export regulations. On the other hand, if the product is such that “strong” cryptography functions are used for both authentication and signature, and encryption is performed with “weak” cryptography functions, it could be exported.⁷ This example shows the need to have different versions of the same product, according to the export regulations. We address this aspect in more details in the following section.

4 What to Do If Your Product Still Falls Into Cryptographic Regulations

The previous design principles help system designers to build smart card based applications meeting the current cryptographic regulations. However, those principles do not ensure that product does not fall into export control. In such a case, we recommend the following design principles.

Principle 7 *Provide mechanisms for variable key lengths and/or key masks.*

As we mentioned it before, using cryptographic functions for authentication and/or digital signature is not subject to cryptographic regulations. Then, having different versions of the same product, according to the cryptographic functions used to implement encryption, is a direct solution to deal with export regulations. Besides, this allows to reduce the time to market: it is possible to deliver a primary product for which the cryptographic functions are “weak”, waiting for the export authorizations.

Since export regulations are related to the key lengths (see §2.2), we identify two approaches to specialize the smart card based product according to the export constraints:

- using variable key lengths for limiting the encryption functions; and
- masking the keys used for restricting the encryption functions.

One should remark that the previous principle has to be applied to the whole environment, and to the development environment in particular. This requires specific solutions that are out of the scope of this paper.

For instance, let us consider a smart card that implements both encryption and digital signature mechanisms. Then, by taking the management of the length of the encryption keys into account at the design stage, the system designers provide the ability to build the following smart cards:

⁷ Of course, the authentication and signature functions should be at least compliant with all the previous principles.

- a smart card for which digital signature is based on “strong” cryptographic functions, but encryption on “weak” cryptographic functions ; and
- a smart card for which both digital signature and encryption are based on “strong” cryptographic functions.

The first type of smart cards will be free regarding export regulations, whereas the second will require export authorization.

Principle 8 *Check the smart card initialization and/or personalization.*

During the initialization or personalization processes, some elementary measures may be taken so as to assist the granting of the export authorizations. For instance, when using a “strong” digital signature algorithm, one can limit the value of the public exponent (e.g., 3, 17 or $2^{16} + 1$). In this way, the signature algorithm can not be used as a “strong” encryption algorithm by swapping public and secret keys.

Concerning key downloading, we recommend a process that permits to write each key once and only once, and preferably during the initialization or personalization process. In this way, the entity in charge of the personalization may guarantee that the smart card given to the issuer cannot be used in an unappropriated way.

5 Conclusion

In this article, we present design principles that help system designers to build smart card based applications meeting the current cryptographic regulation requirements. Although these principles are simple, they are not usually taken into account, and it finally results in increasing drastically the development process of the product.

With the emergence of the common criteria, it seems essential to us to promote the integration of the previous design principles in the development process of the future smart card based products, so as to increase the software and hardware re-uses, and decrease the time to market.

References

1. Wassenaar arrangement site. <http://www.wassenaar.org>.
2. R. Anderson. Crypto in Europe - Markets, Law and Policy. In *Cryptography Policy and Algorithms*, July 1995.
3. W. Diffie and M. E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, November 1976.
4. Electronic Frontier Foundation. *Cracking DES: Secrets of Encryption Research, Wiretap Politics & Chip Design - 1st Edition*. 1998. ISBN 1-56592-520-3.

5. EMV - Europay International S.A., Mastercard International Incorporated, and Visa International Service Association. *Integrated Circuit Card Application: Specification for Payment Systems - Version 3.1.1*, May 1998.
6. Federal Information Processing Standard - Publication 113. *Computer Data Authentication*, 1985.
7. Federal Information Processing Standard - Publication 180-1. *Secure Hash Standard (SHA-1)*, 1995.
8. Federal Information Processing Standard - Publication 46. *Data Encryption Standard*, 1977. revised as FIPS 46-1:1988; FIPS 46-2:1993.
9. Federal Information Processing Standard - Publication 81. *DES Modes of Operation*, 1980.
10. Internet Request for Comments 1321. *The MD5 message-digest algorithm*, April 1992.
11. Internet Request for Comments 1828. *IP Authentication using Keyed MD5*, August 1995.
12. Internet Request for Comments 2104. *HMAC: Keyed-Hashing for Message Authentication*, February 1997.
13. B.-J. Koops. Crypto Law Survey. <http://cwis.kub.nl/~frw/people/koops/law-survey.htm>.
14. V. Miller. Use of Elliptic Curves in Cryptography. In *Advances in Cryptology - CRYPTO'85 Proceedings*, Lecture Notes in Computer Science, pages 417–426. Springer Verlag, August 1985.
15. J. Pieprzyk and B. Sadeghiyan. *Design of Hashing Algorithms*, volume 756 of *Lecture Notes in Computer Science*. Springer Verlag, 1993.
16. R. L. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.