43

A VERY SMART CARD FOR FIGHTING AGAINST TERRORISM[1]

Jean-François Geneste
Astrium. 31, rue des cosmonautes. 31402 Toulouse cedex.

1. INTRODUCTION

On September 11th 2001, 2 planes were flown into the New York Twin Towers while another was flown into the Pentagon. The main problem came from the fact that the terrorists were able to fly the planes without authorization. In [1], the interest of identifying the pilot before obeying his orders has been discussed and **the need for smart cards having no secret inside has been raised (these cards could be stolen or lost with no damage for their security).** One of the main ideas of [1] is to identify in a secure way, the pilot. The case when the legal pilot is the terrorist himself is treated in [1] but is out of the scope of this paper.

Let us therefore now turn to our problem, which is authentication of the legal pilot, and let us examine the characteristics of the cards used today. In fact all the cards store secret keys within their chip because of the protocols they use. These keys are protected by hardware tricks, which are themselves kept secret. We can point out a first inconsistency. At least in theory, the algorithms used by the smart cards can be made public whereas the hardware protection must be kept secret. The second point we want to raise is about the real difficulty of reading the chip of the card or cloning it. It is well

[1] Patent pending

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: 10.1007/978-0-387-35586-3_46

M. A. Ghonaimy et al. (eds.), *Security in the Information Society*

known that it is an easy reach if you have money enough. In fact, no electronic circuit can be made sure under some types of attacks. And in our case we are facing terrorists with unlimited funds. In fact, what we want to avoid is what we call a long-term attack. Namely, let us imagine the pilots own smart cards making it compulsory for them to identify before authorization from the ground of flying a plane, and that one of them (at least) has been stolen his card. Then, can the terrorists get any advantage several years or tens of years later when trying to use the stolen card or trying to defeat the whole identification system?

The answer we propose to give in this paper is that we can ensure a perfect security whatever is the power of the terrorists. For this, we design a card, which can prove it knows a secret but does not have any secret within it. We treat all the cases of attack as well those at the ground level as the ones on board.

The paper is organized as follows. In part 2 we give a general description of the type of PKC we need and we even give the best-suited existing algorithm. In part 3 we explain how we use our scheme. Part 4 is devoted to the description of the card and its associated protocols whereas part 5 consists in a quick (because of a lack o time!) assertion of the security provided.

2. GENERAL DESCRIPTION OF THE PKC

Let (E, ⌡, |, ⌠...) a set where ⌡, |, ⌠... are operations on E. We say that E is a structure. Some well known examples of structures are (∇, +, .), (∧/n∧, .) Etc. Given some properties of a structure, we will say that they allow defining a model of structure. Some well-known models are groups, rings, fields, algebras Etc. But there are also sub models such as unique factorization domains, non-commutative rings and so on.

Conjecture 1

Given a structure model, then there exists at least one problem ÷, which is in NΠ-Π. (We do not make here any difference between probabilistic computations and deterministic ones, but it could be useful in some cases).

For example, a good candidate for ÷ is finding the eigenvalues of an endomorphism over a finite dimensional module over a non-commutative ring.

Now given a structure (E, ⌡, |, ⌠...) it is often possible to find an equivalence relation ~ on the Cartesian product $E_1 \times E_2 \times .. \times E_n$ where the E_i's are subsets of E and where ~ is consistent with the structure of E. That means (F, ⌡, |, ⌠...) has the same structure model as E where F= $E_1 \times E_2 \times .. \times E_n /\sim$. An example of such a construction is when building the ring of fractions of a non-commutative ring with the equivalence relation known as *the right Ore condition.*

Assumption 1

Given (E, ⌡, |, ⌠...) a structure and its associated problem ÷ by conjecture 1, let us assume that there exists over E a PKC f, which is resistant against a chosen cipher text attack. What we mean here is roughly that the cryptosystem is significant in the sense of Bellare [3] for proofs of knowledge. Namely, if a corrupted Alice $\tilde{A}$ can decipher a polynomial number of messages that she is submitted, then there exists a polynomial time Turing machine M, using $\tilde{A}$ as an oracle that can find the secret key of f with overwhelming probability. This significantcy property is easy to be shown equivalent to resistance to a chosen ciphertext attack in the Naor-Yung Model [4]. Roughly speaking, this can be viewed as follows. Let us remember that a PKC f is resistant against a chosen ciphertext attack in the Naor-Yung model when no distinguisher exists able to discriminate 2 cleartexts m_0 and m_1 when given a ciphertext c corresponding to either one of these messages with probability ½. Now let us assume that the real property of our scheme is the following. *Finding one bit of information about the cleartext x, for polynomially many x, given the ciphertext y, on an adaptive chosen ciphertext attack is equivalent to solve the problem ÷ in polynomial time.* This is well known to be equivalent to the Naor-Yung property. Now, let us consider a corrupted Alice $\tilde{A}$ and let us build the following Turing machine M. M has complete control over $\tilde{A}$ and it picks random entries of the form y as potentially valid ciphertexts. Then M outputs $\tilde{A}$'s answer x. What we need to prove is that M knows A's secret[2]. Just let us remark that M can pick a polynomial number of entries y. It gets their deciphering x, polynomially many times. Therefore M is able to decipher a polynomial number of messages thus being able to solve in polynomial time the problem ÷ by resistance to a chosen ciphertext attack. But necessarily, f is based on that problem and f's trapdoor too. Therefore f's trapdoor is available to M.

Q.E.D.

Assumption 2

We want to catch here the notion of intrinsic knowledge an attacker has about the clear text given a chosen cipher text. In fact, even if the algorithm is resistant against a chosen cipher text attack and even if the encipherment is probabilistic, it can happen that, say, by construction, the clear text has some specific properties. For example for a PKC over a module or algebra,

[2] Here we denote $\tilde{A}$ for the corrupted participant whereas we call A the honest player. We have also implicitly assumed that $\tilde{A}$ can decipher polynomially many messages.

the clear text may lie in a certain direction or subring Etc. In the following, we will consider that the PKC we use, f, is what we call a *Public Predicate PKC* (PPPKC). We formally define what it means. Let us call K(x) the *whole* a priori knowledge the attacker has about any clear text x. We assume that this knowledge can be represented by a polynomial time computable predicate P(x, y) where the output of P(x, y) represents the best that can be computed from the knowledge K(x) about the clear text x corresponding to the cipher text y. We shall say that the public predicate is indistinguishable if there exists a polynomial time Turing machine M, which is able to pick at random (i.e. with the uniform distribution) some $x' \neq x$ such that P(x', y). Please notice that there must be at least an $x' \neq x$ otherwise the attacker could get the clear text that way and we would no more have a PKC. We will call such a PKC a PIP PKC for *Public indistinguishable Predicate PKC*. Such a property has been shown to exist in [2].

Assumption 3

We assume we can build as above from E, a structure of the same model, F. Now the algorithm f over E can be turned into the same algorithm over F. We still call it f. We also note $\dot{x}$ the elements of F.

Theorem

Let $\dot{x} \in$ F and $\dot{y} =$ f($\dot{x}$)$\in$ F the corresponding enciphered message. Deciphering $\dot{y}$ gives $\dot{x}=f^{-1}(\dot{y})$. However, an encipherment machine never directly works on equivalence classes but at best on representatives of the equivalence classes. Moreover, the deciphering algorithm is obviously secret so that it can make, internally, some random choices in its computations, leading to get the right result (i.e. $\dot{x}$), but it gets it under the form of a uniformly distributed representative, say x' such that $\dot{x}'=\dot{x}$. Under that condition, and the one that f is a PIP PKC, the following round of communication constitutes a zero-knowledge proof of knowledge in the sense of Fiat-Shamir.

Bob randomly chooses $\dot{x} \in$ F and computes $\dot{y} =$ f($\dot{x}$). Bob sends Alice $\dot{y}$. Alice computes $\dot{x}'=f^{-1}(\dot{y})$ and sends it to Bob. Bob then verifies that $\dot{x}'=\dot{x}$. If this is the case he accepts the proof, otherwise he rejects.

Proof:

The 3 properties we have to prove are consistency, significantcy and zero-knowledge.

Consistency

It is clear that if both Bob and Alice follow the protocol, then the probability of success is 1.

Significantcy

This is our assumption 1 and is equivalent to the fact that f is resistant to a chosen cipher text attack in our model (see [2] for a more complete and example proof).

Zero-knowledge

We now face a corrupted Bob called $\widetilde{B}$. We have to prove that interacting with Alice $\widetilde{B}$ cannot get any bit of information about the secret. Let us therefore consider the following polynomial time Turing machine M. M has complete control over $\widetilde{B}$. Whenever $\widetilde{B}$ asks a question $\dot{y}$ to A (we call $\dot{x}$ the clear text corresponding to $\dot{y}$) M computes $\dot{x}'$ in polynomial time verifying $P(\dot{x}',\dot{y})$ where $\dot{x}'$ is chosen with the uniform probability among the vectors $\dot{x}$ verifying $P(\dot{x},\dot{y})$. For a given $\dot{y}$, $P(\dot{x}',\dot{y})$ represents the *whole* polynomial time computable knowledge about $\dot{x}$. Now we have to prove that the view of M is the same as the view of $\left(A,\widetilde{B}\right)$. However, assumption 2 implies that $\dot{x}'$ is random among the vectors verifying $P(\dot{x},\dot{y})$. On the other hand, A answers $\dot{x}$. Let us then assume that there is a distinguisher T able to distinguish $\dot{x}$ and $\dot{x}'$. Let us consider the following algorithm. Pick a random chosen clear text $\dot{x}$, encrypt it as $\dot{y}=f(\dot{x})$, choose in polynomial time $\dot{x}'$ such that $P(\dot{x}',\dot{y})=P(\dot{x},\dot{y})$. Repeat this polynomially many times. Then feed T with $\left(\dot{x},\dot{x}',\dot{y}\right)$. Then we have a distinguisher for the chosen cipher text attack in the Naor-Yung model, which is in contradiction with assumption 1.

Q.E.D.

We now give an example of such a construction. The interested reader is referred to [2].

Let A be any finite ring (commutative or not) and E = A^3. Let us assume that Bob wants to send a message to Alice, and let us describe how Alice makes her public key. She firstly runs a random generator outputting 3 uniformly chosen values in A, all non-zero and all different, $\lambda_1,\lambda_2 and \lambda_3$. She then picks with the uniform distribution a random 3x3 invertible matrix of the form $h^{-1}=\begin{pmatrix}1 0 0\\ 0 \rho \psi\\ 0 \omega \varphi\end{pmatrix}$. Then Alice's public key is $F=h^{-1}\begin{pmatrix}\lambda_1 0 0\\ 0 \lambda_2 0\\ 0 0 \lambda_3\end{pmatrix}h$.

Encryption algorithm

We assume the message to send is under the form x ∈ E. Bob then picks at random with the uniform distribution 2 other vectors y and z in E such that {x, y, z} is a basis for E. Let us call g^{-1} the matrix made by the coordinates of x, y and z in column and σ the second coordinate of y. Bob then computes $f=g^{-1}Fg$, and σ^2 x + y + z, and sends to Alice the cipher text (f, σ^2 x + y + z).

Decryption algorithm

The decryption algorithm is only a matter of linear algebra and the reader is referred to [2] for the details. It makes essential use of the knowledge of the eigenvalues, which constitute the secret key.

The complete algorithm has been proven to be resistant against a chosen ciphertext attack if A = ∧/n∧ the ring of integers modulo n and if n is chosen like in the RSA [2]. Now it has been conjectured that some other rings that ∧/n∧ are subject to give the same security characteristics, and it was suggested that local rings (commutative or not) could be good candidates. Moreover, always in [2], under the assumption that finding the eigenvalues of a square matrix is a difficult problem, it was proven that if one considers the ring of fractions of a non commutative ring for A, then one is no more able to decipher uniquely. However, deciphering in this case gives, as shown above, a zero-knowledge proof of knowledge protocol in the sense of Fiat and Shamir.

We gave here the minimum to understand the following. We don't want to complexify things, the object being to use the algorithm and not to discuss it.

3. HOW WE USE THE ALGORITHM

In our context, the problem we face is to get a "zero-knowledge proof" protocol involving 3 parties. What we propose is the following. Let us assume, as above, that we have a protocol issued from a PKC resistant against a chosen cipher text attack. Let f be the algorithm. We also suppose that Alice (the pilot!) wants to prove Bob (the plane!) that she is authorized to fly the plane and that Clair (the ground!) is her guaranty. **Now if there is no possible direct interaction between Alice and Clair, let us assume that Alice picks $\dot{x}$ at random (i.e. with the uniform probability) and that she forms the triple (Alice, $\dot{x}$, $\dot{y} = f(\dot{x})$). She then gives her triple to Bob who filters it and sends (Alice, $\dot{y}$) to Clair. Upon reception, Clair deciphers and gets $\dot{x}'=\dot{x}$ and sends back x' to Bob. If $\dot{x}'=\dot{x}$ then Bob accepts the identification otherwise he rejects**. Do we still have a zero-knowledge proof of knowledge? The answer is yes. In fact, the merger of both Alice and Bob, as polynomial time Turing machines still makes a PTM and therefore the protocol is zero-knowledge.

Now we face another problem. As said earlier, we want to achieve a smart card with no secret inside. But as soon as we merge Alice and Bob, the latter knows f and can forge as many identifications as he wants. We see how to solve this problem in the next section.

4. THE CARD AND ITS PROTOCOL

In this section we now build the card step by step. We saw that the knowledge of f by Bob is a big problem. From now on we assume that Alice identifies via her card. Let us then assume that the binary representation of f is $u_1,...,u_n$. Now let's choose some u_i's at random (i.e. with the uniform distribution) and let us force them to zero. We get a new algorithm, f '. Let us store it in Alice's card. Let us call the bits of f forced to zero $v_1,...,v_m$. For example m = 128. Now let us include within Alice's card a random generator which outputs, under Alice's PIN, the sequence $v_1,...,v_m$ and noise otherwise. Let us also assume that there is a battery within the card and a keyboard so that Alice can type her PIN directly on this keyboard with no power supply from the outside. What is the protocol then? **Alice types her PIN and the pseudorandom generator outputs the right sequence allowing to build f from f ' (recall if the typed PIN is not the right one then another sequence is generated giving f"$\neq$f). Then another random generator, within the card, picks a random $\dot{x}$ and forms the triple (Alice, $\dot{x}$, $\dot{y}$ = f($\dot{x}$)) as in the previous section. Now the protocol is the same as in section 3. We only have to care that once the triple has been computed, then a special device inside the card has to erase f in order that only f ' is visible from Bob or anyone else. The computation of the triple must be done in Alice's card and f erased before Alice inserts her card in Bob's device.** We postpone the security proof to the next section to turn to another problem.

Our problem now is to avoid replay of a given transaction. A valid transaction could have been recorded and then been replayed by the terrorists. The model we use is the one of a dishonest Bob. At this step, Bob could record a triple and send it twice to Clair or even more. What we suggest is then to use an extra invertible function, g_d, so that d is the current date and time. Then the protocol is the following. **Alice types her PIN, forms the quadruple $(Alice,\dot{x},\dot{y}=f(g_d(\dot{x})),d)$ and sends it to Bob. The latter filters and sends Clair the triple (Alice, $\dot{y}$, d). Upon reception, Clair verifies if d is consistent with her own date and time. If this is the case, then she computes $\dot{x}'=g_d^{-1}(f^{-1}(\dot{y}))$ and sends it back to Bob. Bob then verifies and accepts or not depending on whether $\dot{x}'=\dot{x}$ or not.** We must add that the data making d must be included in Alice's card in order to be available for the computation of the quadruple in order to allow the erase function for f to work before inserting Alice's card in Bob's reader. See figure 1 for more precisions.

The last problem we face is when Clair is dishonest. The case we treat here is when the ground is under threat of terrorists or if some ground station tries to impersonate Clair. Another example is when Clair would forge false transactions and could deny her responsibility. How can Alice avoid such a fact? The answer is quite easy. Just give Alice, within her card, the possibility to sign with a function h. Then Alice sends Bob the quintuple $(Alice,h(Alice,d),\dot{y},\dot{x},d)$. The protocol does not change, but, in case of any contest on the transactions from Alice, then Clair would have to exhibit h(Alice, d) which she cannot forge by construction of a signature scheme, as a proof of a demand from Alice. We can also remark that in the case we assume the link between Bob and Clair is securized, Clair could have to prove a demand of a transaction by Bob and therefore avoid any signature by Alice.

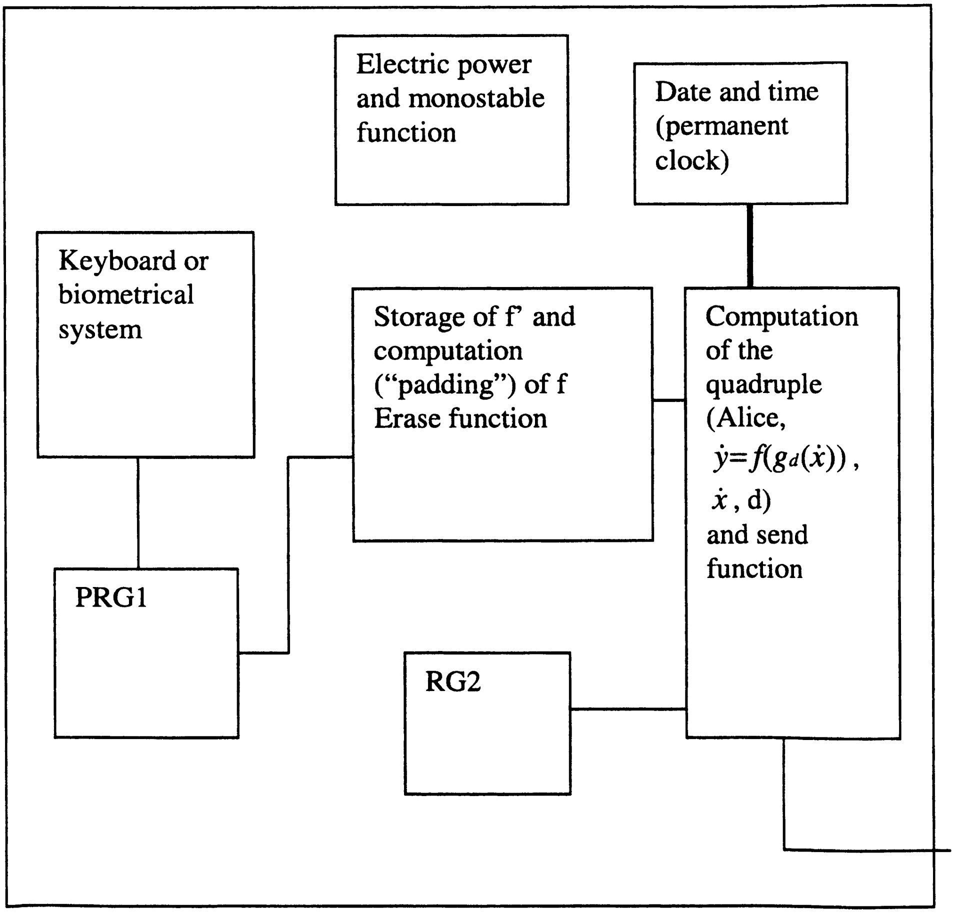
Electric power and monostable function
Date and time (permanent clock)
Keyboard or biometrical system
Storage of f’ and computation (“padding”) of f Erase function
Computation of the quadruple (Alice, $\dot{y}=f(g_d(\dot{x}))$, $\dot{x}$, d) and send function
PRG1
RG2

Figure 1. Schematics of the smart card

5. PROOF OF SECURITY

In the following we prove the security of the whole system. Firstly we list the potential attacks. These are related to the model of Turing machines we work with. This model is described in figure 2.

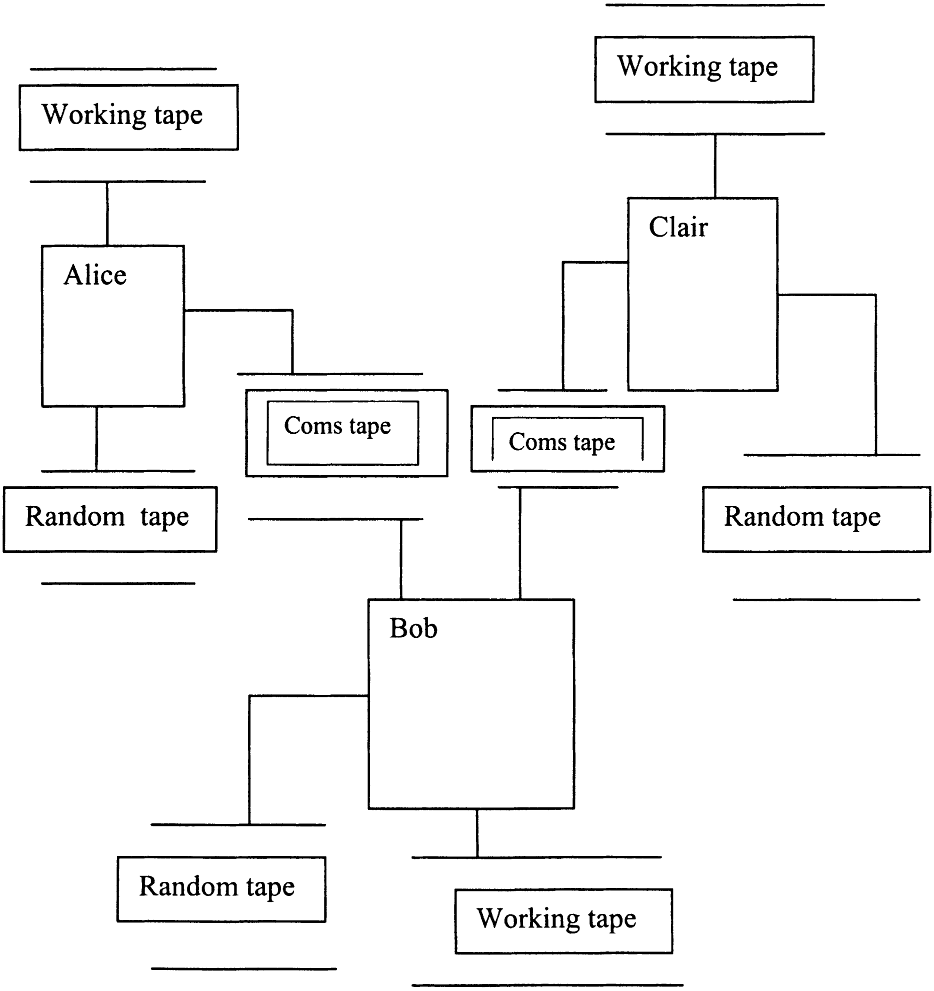

Figure 2. How the Turing machines are communicating

1- The first thing we have to verify is that the proof remains zero-knowledge.

2- Then, is it possible for Bob, in the context of a transaction, to replay Alice's connection after a while, thus asking for the same identification twice or more?

3- Is it finally possible to try a meaningless transaction
in order to
get illegal authorization?

May be there are other possible types of attacks, however, we believe these are the main ones and we are going to answer each of them. But before, let us quickly describe what we think the reality is. Firstly, Alice may want to identify via Clair but be able to repudiate the transaction. Bob can be interested in colluding with Alice to obtain the same result, but also he can try alone to impersonate Alice and get illegal authorization from Clair. Finally, if Clair is dishonest (a ground attack from the terrorists is possible for hijacking with complicities), since we imposed a signature from Alice, this rules out the case.

The 3 parties zero-knowledge

We said earlier that Alice, Bob, or both could try to cheat to get authorization from Clair. Let us consider the 3 cases. Don't forget to look at figure 2 for the model.

In fact if Alice tries to cheat, then just consider the protocol with 2 Turing machines that is A on one side and the couple (B, C) on the other. Then let us apply the zero-knowledge property to the case when there are only 2 parties and we get the result. The fact that, in our model, no collusion is possible between Bob and Clair makes of (B, C) a Polynomial time Turing machine.

If now Bob and Alice collude together, let us consider the couple (A, B) as a single PTM and then apply the result with 2 parties to ((A, B), C) and we get the result. In that case however, Alice could give access Bob to her algorithm f. This is the case when the legal pilot of the company is the terrorist himself. We cannot solve this problem here. See [1] for a solution.

Finally, if we consider only B cheating, then let us also consider ((A, B), C) and we also get the result.

We went however much too quick in what precedes. We have in fact assumed that the merger of 2 probabilistic polynomial time Turing machines is still a PTM and the power of the merger does not exceed the sum of the powers. But this is not, at least in theory, the case. For example, when merging B and C to obtain (B, C), since C knows f^{-1}, then the new machine (B, C) could generate a valid forged certificate. But it cannot be so since in order the transaction cannot be under contest of Alice, Alice must sign the transaction. We could go on, in detail for every case, but the goal of this paper is only to make the reader feel there is no problem. However, in some cases, there can be sharper points and these are the ones we discuss now.

A particular collusion between Alice and Bob

The case we treat here is the collusion when Alice gives internal access to the chip of her card when she types her PIN. In that case Bob could have access to f ' and f and to the keys of the 2 random generators. Bob could then imagine some false transactions while Alice could be able to prove she was not present at the time of the transaction. It is clear, unfortunately, that if Bob has the same knowledge as Alice, then he can impersonate. However, we propose the following. We said in the description that only after the computation of the ciphertext Alice can insert her card in the reader. In order to prevent Bob from reading the remainder, we propose to erase the random generators tracks as soon as they have been used and that any reading is possible only after the computation of the cipher text is made. We suggest detecting any early connection of Bob's reader with the chip and executing a global erase of the data in that case, thus forbidding the use of these data. We cannot (and we regret it) fight against an intrusive look of what happens in the chip of the card. However, we just remark we are in a very special case of fraudulus use where Alice and Bob collude, have the means to look within the chip in an intrusive way and where Alice argues she was not present at the time of the transaction. Can this case be encountered in the reality?

Replaying old transactions

The case we look at is when Bob sends an old transaction. As we said earlier, the date should avoid such a transaction to be made since it is part of the message. However, the date comes from Alice's card and cannot be changed by Bob, except, like in the previous case, if Alice and Bob collude and then contest the validity of the date. But since Clair verifies Alice's date d and compares it with her current date, depending on a threshold value, she will accept or not the proposed date d, only for a valid new message. Every event leading to sucking up information illegally from Alice with her agreement seems, a priori, meaningless.

Le collet marseillais

Le collet marseillais consists in putting a special device in front of a reader, which swallows the card after recording the PIN. Although this seems very unlikely, some terrorists could try such an installation in a plane with no much risk. Since our card has its own keyboard, then no recording of the PIN is possible and therefore no further use of the card is possible.

Cloning

If a terrorist steals the card and clones it (it is not very difficult!), then what can he do with it? In fact he only gets f ', and the PRG, but not the PIN. Therefore he cannot do anything. This is the big advantage of online identification. It is not possible to test the clone off line. Therefore, as soon as the robbery has been signaled, Clair can refuse any trial of transaction from the stolen card.

6. CONCLUSION

We have shown how to derive a zero-knowledge protocol from a PKC resistant against a chosen cipher text attack on a structure model. We have shown, from there, using a concrete example, how to use this property to design a smart card, which needs no secret to be stored within the chip of the card. This makes this new card the most secure ever designed. It is even resistant against the French attack called *le collet marseillais*, and cloning. An application to plane security under terrorist attacks has been proposed and detailed. The result is that the only threats remaining are when the legal pilot is the terrorist himself or when both the ground and the pilot are corrupted. Some solutions have been found to this problem but remain out of the scope of this paper.

REFERENCES

[1] EADS internal call for ideas, *safety of planes under a terrorist attack*. September 2001. Classified unpublished document.

[2] Jean-François Geneste, *Better, Faster and Cheaper than the RSA*, presented at Ecole Normale Supérieure, April 2000. Paper available on the web site http://jeanfrancois.geneste.free.fr.

[3] M. Bellare and O. Goldreich, *On Defining Proofs of Knowledge*, Proc. Crypto'92, pp 390-420. Springer-Verlag.

[4] M. Naor and M. Yung, Public-key Cryptosystems Provably Secure against Chosen Ciphertext Attacks.