Proc. Eurocrypt '92, Balatonfüred, 24.-28.5.1992, to appear in LNCS, Springer-Verlag

Attacks on Protocols for Server-Aided RSA Computation

Birgit Pfitzmann
Institut für Informatik
Universität Hildesheim
Marienburger Platz 22
W-3200 Hildesheim, FRG
pfitzb@informatik.uni-hildesheim.de

Michael Waidner
Institut für Rechnerentwurf
und Fehlertoleranz
Universität Karlsruhe, Postfach 6980
W-7500 Karlsruhe 1, FRG
waidner@ira.uka.de

Abstract. On Crypto '88, Matsumoto, Kato, and Imai presented protocols to speed up secret computations with insecure auxiliary devices. The two most important protocols enable a smart card to compute the secret RSA operation faster with the help of a server that is not necessarily trusted by the card holder.

It was stated that if RSA is secure, the protocols could only be broken by exhaustive search in certain spaces. Our main attacks show that much smaller search spaces suffice. These attacks are passive and therefore undetectable.

It was already known that one of the protocols is vulnerable to active attacks. We show that this holds for the other protocol, too. More importantly, we show that our attack may still work if the smart card checks the correctness of the result; this was previously believed to be an easy measure excluding all active attacks.

Finally, we discuss attacks on related protocols.

1 Introduction

1.1 The Model

Smart cards are often considered as appropriate for carrying out secret cryptographic computations for individual owners. ISO standards for smart cards, however, emphasize flexibility in the physical sense more than flexibility regarding computations. Only rather small chips can therefore be used. Hence, the computing abilities are limited. In particular, at least if the smart card is equipped with a general-purpose CPU, the speed does not suffice for asymmetric algorithms, such as signing a message with RSA [RSA_78]. (Special-purpose designs exist nowadays [QuWB_91, WaQu_91]. Nevertheless, the protocols considered in the following are still to be used in practical systems [KaSh_90].)

In most applications, the smart card communicates directly with a device with much larger computing abilities, such as a point-of-sale terminal. Such a device will be called a *server* in the following. The basic idea of [MaKI_90] (and previous Japanese publications by the same authors) was to use the computing power of the server to help the smart card. This is complicated by the fact that the owner of the smart card need not trust the server.

The question whether an untrusted server can help a less powerful device with a secret computation can be seen as a general theoretical problem, too. A similar problem has been considered in [Feig_86, AbFK_89]. There, the server has unrestricted computational power and needs this power even in the correct protocol. In contrast, here the server is restricted to polynomial-time computations both in the correct protocol and in the attacks. (Otherwise, it could break RSA anyway.)

1.2 Overview over Protocols and Attacks

The main existing server-aided protocols enable a smart card to compute RSA signatures faster with the help of the server. They all share the same basic structure. In Section 2, we describe the first two protocols, RSA-S1 and RSA-S2 from [MaKI_90], and sketch the remaining ones [MaKI_90, QuSo_91, LaYH_91]. The performance of some protocols has been further considered in [KaSh_90].

In [MaKI_90], active attacks are not considered, and it is claimed that if RSA is secure, the best possible passive attacks are brute force search in certain search spaces. We will show that this is not correct, and that much smaller search spaces are sufficient. These attacks can be countered by increasing the system parameters. However, one must carefully consider at what point one loses the advantage over direct computation.

The attacks also work for all protocol variants except that from [QuSo_91], which is provably secure against passive attacks if RSA is secure. However, its drawback is large communication overhead, which makes it impractical for smart cards with a standard interface.

It had already been noticed that the scheme RSA-S2 is vulnerable to active attacks, see a remark in [QuSo_91] and a complete description in [ShKa_90].[1] In both cases, it is proposed that the smart card should check the result, i.e., the RSA signature, before it outputs it to the server. This is possible if the public RSA exponent is small. It is claimed in [ShKa_90] that this countermeasure excludes all possible active attacks. However, we establish a new active attack that requires stronger countermeasures. Additionally, the new attack also works for RSA-S1 and the protocol from [QuSo_91].

We also show that another protocol from [MaKI_90], used to solve modular equations, is not secure. Finally, we make some remarks about another protocol in [QuSo_91].

Note that server-aided protocols for testing RSA signatures with small public exponent exist, too [QuSo_91, Bos_92].

1.3 Other Security Considerations

Note that in the example of ISO standard smart cards, the basic assumptions of the protocols considered here are a little inconsistent: Such smart cards have neither a keyboard nor a display. Consequently, the owner enters secret data, such as a PIN, via the server, which is considered as insecure, and the owner cannot check that the correct message is signed. Thus, no real security can be achieved in this scenario. (And if one deviates from the ISO standards by adding keyboards and displays, one can also use devices with more computational power, e.g., [PrCh_89, BaEi_90].)

---

[1] Two further references containing active attacks and countermeasures, which we unfortunately cannot read, are: Tsutomu Matsumoto, Hideki Imai: How to Ask and Verify Oracles for Speeding Up Secret Computations, Part 1 and 2 (in Japanese); IEICE Technical Reports (Institute of Electronics, Information and Communication Engineers) 89/45 (1989) 21-28, ISEC89-4, and 89/145 (1989) 13-20, IT89-24.

Remark: It has been argued that one might build in a display without deviating from the standard otherwise, and solve the PIN problem with measures such as in [MaIm_91]. Although this approach is interesting (in particular against the threat that someone is watched while typing a PIN), its effectiveness against repeated use of a smart card with the same server, with parameter sizes acceptable for human users, remains to be shown.

2 The RSA Protocols

The smart card wants to compute $y = x^d \bmod n$, where n is the product of two large primes p and q and d is a secret exponent. Both basic schemes, RSA-S1 and RSA-S2, have two parameters, M and L.

2.1 RSA-S1

As a precomputation for all following signatures, the smart card (or a trusted larger device) breaks down the secret exponent d: It randomly generates an integer vector $D = (d_1,\ldots, d_M)$ and a binary vector $F = (f_1,\ldots,f_M)$ with

$$d = f_1 d_1 + \ldots + f_M d_M \bmod \lambda(n) \qquad (1)$$

and $Weight(F) \le L$. Here, *Weight* denotes the Hamming weight. The computation of the signature on an actual message x proceeds as follows:

1. The smart card sends n, D, and x to the server.
2. For $i := 1,\ldots, M$, the server computes
$$z_i := x^{d_i} \bmod n$$
   and sends $Z := (z_1,\ldots, z_M)$ back to the smart card.
3. The smart cards obtains y by multiplying the z_i's for which $f_i = 1$.

Thus, the smart card only needs $L - 1$ multiplications, and the communication is $2(M + 1)$ numbers.

2.2 RSA-S2

RSA-S2 is to improve on RSA-S1 by use of the Chinese remainder theorem, similar to [QuCo_82]. As a precomputation, the smart card breaks down d as

$$d = f_1 d_1 + \ldots + f_M d_M \bmod (p-1)$$

and
$$d = g_1 d_1 + \ldots + g_M d_M \bmod (q-1),$$

where $D = (d_1,\ldots, d_M)$ is an integer vector again, and $F = (f_1,\ldots,f_M)$ and $G = (g_1,\ldots, g_M)$ are binary vectors with $Weight(F) + Weight(G) \le L$.

Steps 1 and 2 are just like in RSA-S1.

In Step 3, the smart card obtains $y_p := y \bmod p$ as the product of the z_i's for which $f_i = 1$, and $y_q := y \bmod q$ as the product of the z_i's for which $g_i = 1$. Finally, it applies the Chinese remainder theorem.

2.3 Other Variants

First, one can also use the Chinese remainder theorem to speed up RSA-S1: The computation is exactly like that in RSA-S2 when F and G are equal. However, the security considerations are different for the two versions. Anyway, since the difference to RSA-S1 is only in the local computations of the smart card, the scheme is just as secure as RSA-S1.

So-called non-binary variants of RSA-S1 and RSA-S2 are obtained if the coefficients f_i and g_i may have other values than just 0 and 1 [MaKI_90, KaSh_90]. Of course, the values must still be very small integers so that the smart card needs just a few multiplications.

The remaining protocols vary the choice of the set of exponents, D:

In [LaYH_91], D is chosen so that the server can compute all the powers $x^{d_i} \bmod n$ more easily with one addition chain. (This is the second proposal in that paper; the first one only changes the local computation of the server.)

Last but not least, the only variant which makes a real security difference is that from [QuSo_91]: There, one and the same fixed set D is used for all smart cards. This scheme is obviously secure against passive attacks, i.e., attacks where the server always sends back correct data: The only information that the smart card gives the server is n, x, and the signature, just as if it computed the signature alone. Unfortunately, a set D which allows every possible secret exponent d to be expressed as in Formula (1) with a vector F of small weight is larger than special sets for special exponents. This increases the communication overhead. In the example in [QuSo_91], $|D| = 832$ for 512-bit exponents. With a standard ISO interface, i.e., 9600 bit/s, the communication would take more than 40 seconds. Hence, in spite of its security advantage, this variant cannot be used in several applications, and it still makes sense to consider the other variants further.

3 Passive Attacks

By "passive attacks" we mean attacks where the server never deviates from its protocol, i.e., it sends back the correct powers $x^{d_i} \bmod n$. Hence, no measures to prevent passive attacks are possible.

3.1 Passive Attack on RSA-S1

It is stated in [MaKI_90] that RSA-S1 could only be broken by searching the true value d via the exhaustion of

$$\sum_{i=1}^{L} \binom{M}{i} \tag{2}$$

possibilities. (That is, all the possible vectors F of weight $\leq L$.)

The following attack shows that the search space can be considerably smaller:

(1) For a message x where the signature $y = x^d$ is known, one first computes all the values $z_i = x^{d_i}$.

(2) Next, one computes all the products

$$y_F := \prod_{i=1}^{M} z_i^{f_i} \bmod n$$

for vectors F of only half the weight, i.e., with $Weight(F) \leq \lceil L/2 \rceil$.

(3) One also computes the values

$$y^*_F := y\, y_F^{-1} \bmod n$$

for the same vectors F.

(4) The values y_F are sorted, and the values y^*_F compared to them. Whenever a match is found, i.e., $y_{F_1} = y^*_{F_2}$, and F_1 and F_2 are disjoint (i.e., they do not have a 1 at the same position), the vector $F_1 + F_2$ is a candidate for the true secret vector F.

(5) If F is not uniquely determined by this procedure, one can test the remaining candidates by use of a second message x'.

The reason why this works is that the true vector F can be represented as the sum $F_1 + F_2$ of two disjoint vectors of weight $\leq \lceil L/2 \rceil$. The equation for the signature is

$$y = \prod_{i=1}^{M} z_i^{f_i} = \prod_{i=1}^{M} z_i^{f_{1,i} + f_{2,i}} = y_{F_1} y_{F_2} \bmod n.$$

Thus
$$y_{F_1} = y\, y_{F_2}^{-1} = y^*_{F_2} \bmod n.$$

Complexity. The number of values y_F is

$$N := \sum_{i=0}^{\lceil L/2 \rceil} \binom{M}{i},$$

and they can be computed with little more than one multiplication on average (if one stores intermediate results). The same holds for the values y^*_F, if one starts by computing the values z_i^{-1}. Sorting and searching take about $N \log(N)$ operations.

All this is considerably less than the number in Formula (2) (a bit larger than the square root).

3.2 Passive Attack on RSA-S2

It is said in [MaKI_90] that RSA-S2 could only be broken by searching the true value d via the exhaustion of

$$\sum_{j=1}^{L} \sum_{i=0}^{j} \binom{M}{i}\binom{M}{j-i} > \binom{M}{L/2}^2 \tag{3}$$

possibilities. (That is, all combinations of vectors F and G of total weight $\leq L$.) How two vectors are actually checked for correctness is described in [KaSh_90].

Again, we can reduce the search space considerably: One can search for one of the vectors F or G individually. Clearly, one of them must be of weight $\leq L/2$.

(1) For a message x where the signature $y = x^d$ is known, one first computes all the values $z_i = x^{d_i}$.

(2) Next, one computes all the products

$$y_F := \prod_{i=1}^{M} z_i^{f_i} \bmod n$$

for vectors F with $Weight(F) \leq L/2$.

(3) One also computes the values $v_F := \gcd(y_F - y, n)$. If one of them is neither 1 nor n, one has factored the modulus.

(4) Otherwise, at least one value v_F (and usually not many) will be n, and either the true F or the true G must be among the corresponding vectors. Normally, one will find $F = G$ in this case.

The reason why this works is that for the true F, $y_F = y_p = y \bmod p$, and for the true G, $y_G = y_q = y \bmod q$. Thus, not all the values v_F can be 1. Furthermore, except when $F = G$, y_F will usually not be congruent $y \bmod q$, too, and vice versa. In these cases, one can factor the modulus according to Step (3). Otherwise, one has usually found both F and G. In the remaining cases, a very small search space will remain, and anyway, it is enough for an attack to succeed in most cases.

Complexity. The number of values y_F is

$$N := \sum_{i=0}^{L/2} \binom{M}{i},$$

and, as above, they can be computed with a bit more than one multiplication on average. Since the greatest common divisor takes more time, one should initially compute it for the product of several differences $(y_F - y)$.

Again, the complexity is not much more than the square root of that described in Formula (3).

3.3 Passive Attacks on Other Variants

The attacks described easily generalize to the non-binary variants of RSA-S1 and RSA-S2. Furthermore, the protocol from [LaYH_91] is just a special case of the original protocols as far as security is concerned. Hence the attacks work for all known variants except the provably secure one from [QuSo_91].

4 Active Attacks

In active attacks, the server deviates from its protocol by sending back a wrong vector $Z' = (z'_1,\ldots, z'_M)$, instead of the powers $z_i = x^{d_i} \bmod n$. On the one hand, this makes active attacks more powerful, and they usually result in a total break of the system in a few steps. On the other hand, active attacks can often be detected or even prevented, in contrast to passive ones. Hence, the most important thing to know about active attacks is not how they work, but whether one exists and which countermeasures are effective.

4.1 Description of Attacks

Attacks on RSA-S1. As mentioned, we describe the first active attack on RSA-S1. The basic idea is to use Jacobi symbols: If the server sends back any vector $Z' = (z'_1,\ldots, z'_M)$, then the smart card outputs

$$y' := \prod_{i=1}^{M} z'_i{}^{f_i} \mod n.$$

The server can compute Jacobi symbols modulo n. Let $(-1)^c$ be that of y', and let I be the subset of indices i where the Jacobi symbols of z'_i are -1. Since Jacobi symbols are multiplicative, the server knows

$$(-1)^c = \left(\frac{y'}{n}\right) = \prod_{i=1}^{M} \left(\frac{z'_i}{n}\right)^{f_i} = \prod_{i\in I} (-1)^{f_i}.$$

This yields $$c = \sum_{i \in I} f_i \bmod 2.$$

Thus from each vector Z', the server obtains one linear equation about F in GF(2). As long as the server sends back correct data, this does not matter much, since it always obtains the same equation. However, in an active attack, it can choose just one z'_i with Jacobi symbol –1, and the rest with Jacobi symbol +1. Then it obtains the value of f_i directly. Thus, after M rounds of this attack, the server knows F and therefore d.

If L is much smaller than M, the server needs less than M rounds if it chooses the vectors Z so that the resulting linear equations form the parity check matrix of a code correcting up to L errors.

If the server is not likely to meet the same smart card often enough, it can also use the information obtained from some rounds of the active attack to speed up the passive attack.

A far more elegant attack has been found recently [Ande_92]: The server sends back a selection of small primes (or a blinded version thereof), factors the product that the smart card outputs (which is $< n$ if M is not too large) by trial division, and thus finds out F. However (see Section 4.2), this elegant attack is easily prevented by countermeasures that were previously proposed for other protocol variants, and which are needed against our attack, too, whereas our attack needs additional countermeasures. Hence, in practice, the additional elegance makes no difference, and the less elegant attack is even more dangerous.

The Attack on RSA-S2. The attack on RSA-S2 from [ShKa_90] uses the following fact: If the server changes the sign of just one of the values z_i, and if $f_i \neq g_i$, then the resulting value y' and the true signature y are significantly different square roots of the same value. The server does not know y. However, since the public exponent e is odd, the same holds for the values y'^e and $y^e = M$. In this case, the server can therefore factor the modulus by computing $\gcd(y'^e - y^e, n)$.

4.2 Discussion of Countermeasures

Why No Active Attack Can Be Ignored. The attitude towards active attacks in [MaKI_90] was to assume that the server would refrain from them. However, such an assumption about an untrusted server is unjustified unless such an attack would at least entail a severe risk. One risk might be that the owner of the smart card notices that the smart card outputs a wrong signature. However, if the owner obtains the signature at all, it obtains it through the server, and all the attacks described above have a variant where the server can output the correct signature.

With the one-round attacks [ShKa_90, Ande_92], this is clear since the server obtains the secret key at once. With the Jacobi symbol attack, the server chooses a with Jacobi symbol –1 modulo n. It first computes the correct vector Z and its Jacobi symbols. Where it wants the Jacobi symbol changed, it uses

$$z'_i := a\, z_i, \text{ otherwise } z'_i := z_i.$$

Thus the smart card's output y' is the product of the real signature y and a value a^x, where $x \leq L$ and $x \leq$ the number of factors a used. The server can find out the correct signature by searching among the few numbers $y'\, a^{-x}$. If the Jacobi symbol of –1 modulo n is –1, one would use $a = -1$. Then the signature y is $\pm y'$.

Hence each of the attacks implies that countermeasures in the smart card itself are needed.

How Much Does Signature Checking Help? The easiest countermeasure is that the smart card tests the resulting signature and only outputs it if it is correct [QuSo_91]. This restricts the protocol to RSA with small public exponents. It also means additional computation; however, this may be small compared to the L multiplications needed anyway.

In fact, this measure effectively excludes all one-round attacks, such as [ShKa_90, Ande_92].

It was even said in [ShKa_90] that it excludes all conceivable active attacks. However, multi-round attacks are still possible.

For instance, if the server chooses just one z'_i wrong each time, like in the Jacobi symbol attack, it can infer f_i from whether it receives a signature or not. By choosing all the values z'_i as $\pm z_i$, the server can obtain any linear equation about F again. Hence also the variant with error-correcting codes still works.

With RSA-S2, choosing one z'_i wrong reveals $(f_i \vee g_i)$, and changing more than one sign reveals different information.

Other Countermeasures. The only general countermeasure that would certainly exclude all active attacks would be to check the values sent by the server, instead of the result. However, this looks infeasible.

The next-best general solution (from a theoretical point of view) is that the smart card stops once and for all if it detects an attack. However, this may have practical disadvantages. In particular, since the smart card does not have a display, the client will only notice this fact during the next transaction with an honest server.

Instead, one could let the smart card continue and try to issue a warning through the next server it communicates with. However, this seems quite dangerous since someone might do all their shopping in one supermarket for quite a while.

In the special protocols considered here, an easier measure might be to change the vectors D and F with each signature (in addition to checking the result). In this case, the smart card must be able to generate random numbers quickly, and the procedure of breaking down d must definitely be implemented on the smart card itself. Then just one bit of information can be obtained about each F; this corresponds to a passive attack with slightly smaller parameters L and M. However, it is not clear if having several vectors D allows new passive attacks. Anyway, this measure seems impossible with the protocol from [QuSo_91].

5 Attacks on Related Protocols

There are two more types of server-aided secret computations in [MaKI_90].

The first type is matrix multiplication, where the matrices are to be kept secret. In the protocol, the server receives versions of the matrices where the rows and columns are permuted. It has been noticed in [MaKI_90] itself that quite a lot of information, such as the determinants, is not hidden by these operations. Thus this protocol does not provide secrecy in the cryptographic sense.

The second type is solving modular equations. The smart card has secret integers $a_0,\ldots, a_{m-1}$ and an integer k. (k was declared secret, too, but that must have been a slip of the pen.) It wants to know a solution x to the equation

$$a_0 + a_1x + \ldots + a_{m-1}x^{m-1} + x^m = 0 \bmod k.$$

For this, it chooses a random number r and computes $b_{m-i} := a_{m-i}\, r^i$ for $i := 1,\ldots, m$, and sends k and the tuple $B = (b_0, b_1,\ldots, b_{m-1})$ to the server. The server computes a solution y to the equation with the coefficients B. The smart card can compute x from y as $x = y\, r^{-1}$ mod k. This protocol is said to reveal nothing about the secret to the server. It does, however. For example, if $\{x_1,\ldots, x_u\}$ is the complete set of zeroes of the original polynomial, the server can compute $\{r\, x_1,\ldots, r\, x_u\}$and therefore all the quotients $x_i\, x_j^{-1}$. As more easily computable functions of the original secret coefficients, the server can compute $a_{m-1}{}^i\, a_{m-i}^{-1} = b_{m-1}{}^i\, b_{m-i}^{-1}$.

In [QuSo_91], a special protocol for deciphering RSA-encrypted messages is also contained. First, this protocol assumes that the modulus n is secret from the server, which seems a rather strange assumption to make with a public-key cryptosystem. Secondly, if the server receives the decrypted messages (which is not as natural as that the server obtains the signatures, though), this scheme is vulnerable to active attacks, too: The smart card has chosen two additional primes r_1, r_2 and computed $n_1 := pr_1$ and $n_2 := qr_2$. It has also blinded the secret exponent as $\sigma_1 := d_1 + \rho_1(p - 1)$ and $\sigma_2 := d_2 + \rho_2(q - 1)$, where d_1 and d_2 must be the reductions of d modulo $p - 1$ and $q - 1$, resp., and $\rho_1 \le p - 2$ and $\rho_2 \le q - 2$ are random numbers. Now, together with a ciphertext C, it sends n_1, n_2, σ_1, and σ_2 to the server. The server should answer with $M_1 := C^{\sigma_1}$ mod n_1 and $M_2 := C^{\sigma_2}$ mod n_2. The smart card computes the message M by applying the Chinese remainder theorem to M_1 mod p and M_2 mod q. If the server gives back the same value M_1 in two different protocol executions and receives the two results M and M', then $M - M'$ is a multiple of p. Thus, with high probability, $\gcd(n_1, M - M') = p$.

6 Conclusion

We have described several attacks on server-aided computation protocols, in particular, protocols for the computation of RSA signatures. Several of these attacks were previously declared impossible. None of the attacks on the signature protocols is disastrous, i.e., they can all be rendered ineffective by increasing the parameters or by performing additional tests. However, all these countermeasures cost time and may therefore annihilate the advantages of the server-aided approach.

Furthermore, we are by no means sure that our passive attacks are already optimal. In particular, one could try to exploit the obvious connection to modular knapsacks. (In contrast, better active attacks are of no practical importance, since the countermeasures needed so far exclude active attacks generally.)

Thus, like with all unproven cryptographic schemes, one should let server-aided computation undergo a lengthy evaluation phase. In this special case, the result is likely to be that by the time that the schemes are sufficiently well evaluated, smart cards that can compute RSA on their own are available for everyone. Other applications for these or similar protocols are not inconceivable, but one trades computation for communication.

Acknowledgement

We would like to thank Prof. Tsutomu Matsumoto for helping us to more literature about this subject, and Ross Anderson for an interesting discussion.

References

AbFK_89 Martin Abadi, Joan Feigenbaum, Joe Kilian: On Hiding Information from an Oracle; Journal of Computer and System Sciences 39/1 (1989) 21-50.

Ande_92 Ross Anderson: Personal communication, 26.5.1992; to be submitted to Electronics Letters.

BaEi_90 Paul Barrett, Raymund Eisele: The smart diskette – A universal user token and personal crypto-engine; Crypto ’89, LNCS 435, Springer-Verlag, Heidelberg 1990, 74-79.

Bos_92 Jurjen Bos: Practical Privacy; Proefschrift, Technische Universiteit Eindhoven 1992.

Feig_86 Joan Feigenbaum: Encrypting Problem Instances, Or ..., Can You Take Advantage of Someone Without Having to Trust Him?; Crypto ’85, LNCS 218, Springer-Verlag, Berlin 1986, 477-488.

KaSh_90 Shin-ichi Kawamura, Atsushi Shimbo: Performance Analysis of Server-Aided Secret Computation Protocols for the RSA Cryptosystem; The Transactions of The Institute of Electronics, Information and Communication Engineers IEICE, E73/7 (1990) 1073-1080.

LaYH_91 Chi-Sung Laih, Sung-Ming Yen, Lein Harn: Two Efficient Server-Aided Secret Computation Protocols Based on the Addition Sequence; Asiacrypt ’91 – Abstracts, 270-274.

MaIm_91 Tsutomu Matsumoto, Hideki Imai: Human Identification Through Insecure Channel; Eurocrypt '91, LNCS 547, Springer-Verlag, Berlin 1991, 409-421.

MaKI_90 Tsutomu Matsumoto, Koki Kato, Hideki Imai: Speeding up Secret Computations with Insecure Auxiliary Devices; Crypto ’88, LNCS 403, Springer-Verlag, Berlin 1990, 497-506.

PrCh_89 Wyn L. Price, Bernard Chorley: The Intelligent Token or ‘Super-Smart’ Card; SMART CARD 2000 (1987), North-Holland, Amsterdam 1989, 133-138.

QuCo_82 Jean-Jaques Quisquater, C. Couvreur: Fast Decipherment Algorithm for RSA Public-Key Cryptosystem; Electronics Letters 18/21 (1982) 905-907.

QuSo_91 Jean-Jaques Quisquater, Marijke De Soete: Speeding up Smart Card RSA Computation with Insecure Coprocessors; Proceedings Smart Cards 2000 (1989), North-Holland, Amsterdam 1991, 191-197.

QuWB_91 Jean-Jaques Quisquater, Dominique de Waleffe, Jean-Pierre Bournas: Corsair: A chip card with fast RSA capability; Proceedings Smart Cards 2000 (1989), North-Holland, Amsterdam 1991, 199-206.

RSA_78 Ronald L. Rivest, Adi Shamir, Leonard Adleman: A Method for Obtaining Digital Signatures and Public-Key Cryptosystems; Communications of the ACM 21/2 (1978) 120-126, reprinted: 26/1 (1983) 96-99.

ShKa_90 Atsushi Shimbo, Shin-ichi Kawamura: Factorisation attack on certain server-aided computation protocols for the RSA secret transformation; Electronics Letters 26/17 (1990) 1387-1388.

WaQu_91 Dominique de Waleffe, Jean-Jaques Quisquater: CORSAIR: A Smart Card for Public Key Cryptosystems; Crypto ’90, LNCS 537, Springer-Verlag, Berlin 1991, 502-513.