Cryptanalysis of the OTM signature scheme from FC'02

Jacques Stern[1] and Julien P. Stern[2]

[1] Dépt d'Informatique, Ecole normale supérieure, 45 rue d'Ulm, 75230 Paris Cedex 05, France.
E-mail: Jacques.Stern@ens.fr

[2] Cryptolog International SAS, 16–18 rue Vulpian, 75013 Paris, France.
E-mail: julien@cryptolog.com

March 30, 2003

Abstract. At Financial Cryptography 02, Okamoto, Tada, and Miyagi [8] proposed a new fast signature scheme of the Schorr/DSS family, without on line multiplication. Following earlier proposals [5, 10, 11], a part of the data, independent of the message to sign, is generated at a preprocessing stage, while the computing effort needed to complete the signature "on the fly", is dramatically reduced. Whereas the so-called GPS scheme from [5, 10] and its variant from [11] avoid modular operations by computing over the integers, thus reducing the workload to one (regular) multiplication, the new scheme simply gives up multiplication at the cost of bringing back a single modular reduction with respect to a 160 bit integer. Thus, the scheme could appear as achieving better performances. Unfortunately, due to a concealed design weakness, the scheme in [8] is insecure with the proposed parameters. The present paper shows a devastating attack against the scheme, forging a signature in $\simeq 2^{25}$ operations. The scheme can be rescued in a rather straightforward way by significantly raising the parameters, but this degrades its performances which do not compare anymore favorably to [10]. In place, we suggest to replace modular reduction by another novel operation, which we call *dovetailing*. We argue that this operation can be performed in such an efficient way that it could allow for signing with a memory card, rather than a smart card. This equally applies to GPS but the new scheme is better than GPS in terms of signature size.

1 Introduction

In the last twenty years, public key cryptography has dramatically developed, as a means to offer confidentiality to end users. Popular applications such as SSL are now widely accepted as a simple and cost effective method to protect data obtained from WEB servers. However, in many applications, notably outside the Internet, public key techniques are hampered by the computing cost that they require. Currently, banking cards and mobile phones, almost exclusively make use of conventional symmetric cryptography. This entails serious drawbacks such as the impossibility of producing actual digital signatures, or of achieving off-line authentication without storing secrets in the verifying device. Many potential applications would greatly benefit of the availability of signature and identification schemes that could be executed, say, on a low cost smart card. The Holy Grail here would be the use of memory cards: this would allow to authenticate phone cards or to sign up at toll booths, thus opening a mass market to public key techniques.

In an attempt to find a partial solution to the above, several authors [3, 12, 13] have proposed to break signature generation into two steps

1. a preprocessing step, where data independent of the message to sign are generated,
2. a second step, performed "on the fly", where the signature is completed, with limited computing effort.

Even, Goldreich, and Micali [3] have shown a general albeit not very practical method for converting any signature scheme into a scheme of this form. This has recently been pursued by Shamir and Tauman [14]. In a more restricted setting, the discrete logarithm has been found particularly suitable to this approach, as already noted by Schnorr [12, 13]: ignoring one modular addition, the only operation that needs to be performed on the fly is a single modular multiplication. In [5, 10], this was reduced to a single regular multiplication. Besides reducing the computing cost, discarding modular multiplication had several nice consequences in terms of key size or signature size. Aiming at a standard security level measured by 2^{80} basic computing steps, the scheme in [10] allowed for secret keys of 160 bits and for signatures of 420 bits.

A further step was taken in a paper [8] presented at Financial Cryptography 2002. In this paper, a scheme that could sign with no on-the-fly multiplication was proposed. We will refer to this scheme as OTM, from the names of the authors, Takeshi Okamoto, Mitsuru Tada and Atsuko Miyaji. Besides avoiding multiplication, the scheme was achieving secret key size 160 bits and signature size 304 bits, thus clearly beating the GPS scheme from [10], at least in terms of size. The price to pay for giving up multiplication was to bring back a single modular reduction with respect to the secret key s, a 160 bit integer. Whether or not the latter is more time consuming than multiplication is debatable and will be discussed further on. In any case, the answer clearly depends on the size of the integer to be reduced modulo s, and any increase of this size entails a loss of efficiency.

In this paper, we first show that, due to a concealed design weakness, the scheme in [8] is insecure with the proposed parameters. The present paper shows a devastating attack against the scheme, forging a signature in $\simeq 2^{25}$ operations. We next discuss whether the scheme can be rescued. A rather straightforward option consists in significantly raising the parameters. However, as explained above, this degrades the performances which do not any more compare favorably to [10]. In place, we suggest to replace modular reduction by another novel operation, which we call *dovetailing*. Basically, dovetailing a pair of integers (r, e) with respect to a smaller integer s is adding to r a small multiple of s so to as to make the trailing bits of the result match with e. Surprisingly, we found that this operation can be performed in an extremely efficient way. We even argue that the resulting scheme could allow for signing with a memory card, rather than a smart card.

2 A flaw in the OTM scheme

2.1 Brief description of the scheme

The public key of the OTM scheme consists of an RSA integer n, together with an element g of $\mathbb{Z}_n^\star$, whose order is s is bounded by $S = 2^k$ (typically $S = 2^{160}$). Paper [8] describes how to manufacture such pairs (n, g). We will return to the matter further on in the present paper. The secret key is precisely the order s of g. The signature scheme is derived from an identification scheme, depicted on figure 1. Notations a, ℓ refer to additional parameters of the scheme, typically $a = 224$, $\ell = 24$.

Prover		Verifier
	Public data n RSA integer $g \in \mathbb{Z}_n^\star$, k	
$s < 2^k$, $g^s = 1 \bmod n$		
$r \in_R \{r \mid 0 \leq r < 2^a\}$ $x = g^r \bmod n$	$\xrightarrow{x}$	
	$\xleftarrow{e}$	$e \in_R \{e \mid 0 \leq e < 2^{a+\ell}\}$
$y = r + (e \bmod s)$	$\xrightarrow{y}$	
		$y \overset{?}{<} 2^a + 2^k$ $x \overset{?}{=} g^{y-e} \bmod n$

Fig. 1. The OTM identification scheme from FC'02

To derive the signature scheme, one applies the usual trick from [4], replacing e by a hash value $H(m, x)$, computed from the message m and the so-called commitment x. The signature is the pair (e, y). To achieve bit length 304, as claimed in [8], one uses another trick, keeping only say 80 bits of $H(m, x)$ and extending the result to $a + \ell$ bits by means of a hash function again.

2.2 A forgery attack

The security analysis of the OTM appearing in [8], relies on the claim that an attacker cannot cheat the identification protocol with probability significantly larger than $2^{a+\ell}$ (theorem 4 of [8]). This result is incorrect, as shown by the attack that we now mount. The attack is based on guessing the $\ell + 1$ leading bits of the "challenge" e. This means guessing e_1, where

$$e = e_1 2^{a-1} + e_0$$

and $0 \leq e_0 < 2^{a-1}$. From this guess, the attacker manufactures his commitment x by picking at random r, $0 \leq r < 2^{a-1}$ and setting $x = g^{r-e_1 2^{a-1}} \bmod n$. Upon receiving the challenge e, the attacker gives up if the leading bits of e do not match up with e_1. Otherwise, he is successfully identified by returning

$$y = r + e_0$$

Indeed, both tests $y \stackrel{?}{<} 2^a + 2^k$ and $x \stackrel{?}{=} g^{y-e} \bmod n$ are easily seen to be satisfied.

The above directly translates into a forgery attack against the signature scheme. The attacker guesses an $\ell + 1$ bit integer e_1 and computes $H(m_i, x)$ for a sample of messages m_i until the ℓ leading bits of the result match up with e_1. The forgery requires $2^{\ell+1}$ operations.

2.3 Repairing the scheme

It appears that theorem 4 from [8] is correct when the security bound $1/2^b$ is replaced by $1/2^\ell$. Without going into details, we just mention that the "extractor" that the proof builds needs to find a situation where the attacker can answer two queries with different leading bits, given the same commitment. The proof in [8] is erroneous in that it simply looks for distinct queries.

Thus, the signature scheme can be repaired by raising ℓ to at least 80. At this point, we wish to suggest to raise a to at least $k + 80$. The size of a controls the bound $2s/2^a$ for the statistical zero-knowledge property (theorem 5 of [8]). However, this bound is only for a single execution and should be multiplied by the number of identifications (signatures) allowed. Thus, we feel that the suggested margin of 64 bits is not enough.

We have a final concern about the scheme, related to the key generation algorithm. The RSA integer $n = pq$ used in [8] is manufactured in such a way that $p = 2p'p'' + 1$, and $q = 2q'q'' + 1$, with p', q' prime integers and p'', q'' odd. Next, an element g_p of $\mathbb{Z}_p^\star$ of order p' is built, and similarly an element g_q of $\mathbb{Z}_q^\star$ of order $2q'$. Finally g is obtained by Chinese remaindering. Now, the paper states that taking s of size 160 bits is enough to be immune to Pollard's method [9] for finding the order of g, since the method has computing time $\mathcal{O}(\sqrt{s})$. Thus, g has order $< 2^{80}$ in either $\mathbb{Z}_p^\star$ or $\mathbb{Z}_q^\star$. Although we have not found any attack, we are concerned that revealing such a g might open the way to speeding up the factorization of n. Of course, the OTM scheme can perfectly be adapted to the case where n is a prime number. However, in this case, the order of g is a relatively small factor of $n-1$, and accordingly it should be taken large enough to withstand ECM factorization. Given the current ECM factoring record of 54 decimal digits (see [7]), one would have to raise the size of s to at least 256 bits.

2.4 Comparing the resulting scheme to GPS

The GPS scheme from [5, 10] is depicted on figure 2 below.

Prover		Verifier
	Public data n RSA integer $g, v \in \mathbb{Z}_n^\star$, k	
$s < 2^k$, $g^s v = 1 \bmod n$		
$r \in_R \{r \mid 0 \leq r < 2^a\}$ $x = g^r \bmod n$	$\xrightarrow{\ x\ }$	
	$\xleftarrow{\ e\ }$	$e \in_R \{e \mid 0 \leq e < 2^{a+\ell}\}$
$y = r + se$	$\xrightarrow{\ y\ }$	
		$y \overset{?}{<} 2^a + 2^{k+\ell}$ $x \overset{?}{=} g^y v^e \bmod n$

Fig. 2. The GPS identification scheme

Typical parameters in signature mode, where e is replaced by a hash value $H(m, x)$, are $k = 180$, $\ell = 80$, $a = 340$, which yield signature size over 400 bits.

As noticed by the authors of [8], raising the value of ℓ in the OTM scheme has no consequences on the size of signatures, since the challenge e is computed from a short seed anyway (say 80 bits). Thus, even taking into account the slight increase of a that was suggested above, the scheme still beats GPS in terms of signature size, since it achieves bit size 320 whereas GPS stands over 400 bits.

On the other hand, we claim that, in terms of efficiency, the OTM scheme now has much worse performance, when aiming at a comparable security level 2^{80}. Indeed, the core operation that GPS performs on the fly, is the (regular) multiplication of the secret key s (160 bits) by the challenge e (80 bits). By the usual shift and add algorithm, this means 40 additions of a 160 bit integer. This computation sums up to about 200 8-bits multiplication or about 800 8-bit additions, if we are in a multiplication free context.

Let us now look at the core operation involved in OTM in more detail. This operation is the modular reduction of the random challenge e by the secret key s. With the new security parameters suggested above, e is 320 bits and s is 160 bits. We note that s and e must be chosen at random to maintain security, and therefore that they cannot be chosen of a certain specific form that could speed up modular reductions.

Hence, the efficiency of OTM is essentially the efficiency of general modular reductions of 320 bit numbers by 160 bit numbers. The modular reduction primitive on large numbers has been extensively studied in [1], where the authors compare the performances of the three main reduction techniques: the so-called "classical" one, Barrett's algorithm and Montgomery's algorithm.

It turns out, both from their theoretical analysis and from their implementation on a 80386, that the reduction of a $(2k)$-bit number by a k-bit number is roughly comparable to the multiplication of 2 k-bit numbers, with an advantage for the multiplication (it costs between 420 and 480 8-bit multiplications, if we neglect the precalculation and postcalculation steps).

Thus, with the corrected security parameters, the core modular reduction needed by OTM is more than twice more expensive as the core multiplication needed by GPS.

3 A signature scheme based on dovetailing

In this section, we suggest to replace modular reduction by another novel operation, which we call *dovetailing*.

3.1 Dovetailing

Let s be an odd integer. Let (r, e) be a pair of integers, $r > s$, $r > e$. Dovetailing (r, e) with respect to s is adding to r a small multiple of s so to as to make the trailing bits of the result match with e. In mathematical terms, the operation might look intricate: setting $y = r + \lambda s$, one can choose λ as $(e - r)s^{-1} \bmod 2^k$, where k is the bit size of e. However, as will be seen in the sequel, it can be performed by a surprisingly efficient algorithm. We let $\mathcal{D}_s(r, e)$ be the result of dovetailing.

3.2 The new scheme

We now introduce a scheme that uses dovetailing in place of modular reduction. The setting is essentially identical to OTM, except that we need to assume that the order s of g is odd.

The security analysis is similar to the (corrected) analysis of OTM: probability of forgery is of the order $1/2^\ell$ and statistical zero knowledge of a single round of identification is controlled by $2s/2^{a-k-\ell}$. Thus, we suggest to set $k = 160$, $a = 320$, $\ell = 80$.

The signature scheme is derived as usual. Note that, since the trailing bits of y match up with e, it is enough to output y, which means 320 bits for the suggested parameters.

Prover		Verifier
	Public data	
	n RSA integer	
	$g \in \mathbb{Z}_n^\star$, k	
$s < 2^k$ odd, $g^s = 1 \bmod n$		
$r \in_R \{r \mid 0 \leq r < 2^a\}$		
$x = g^r \bmod n$	$\xrightarrow{x}$	
	$\xleftarrow{e}$	$e \in_R \{e \mid 0 \leq e < 2^\ell\}$
$y = \mathcal{D}_s(r, e)$	$\xrightarrow{y}$	
		$y \overset{?}{<} 2^a + 2^{k+\ell}$
		$x \overset{?}{=} g^y \bmod n$
		$e \overset{?}{=} y \bmod 2^\ell$

Fig. 3. An identification scheme based on dovetailing

It should also be noted that checking the size of the response, (e.g. checking that $y \overset{?}{<} 2^a + 2^{k+\ell}$) is not critical for the security of our scheme but is preferable for implementation reasons.

3.3 Implementation

While general modular reductions are fairly complex algorithms, the special case of dovetailing allows for very efficient implementations.

Software implementation We will first show that dovetailing can be implemented in software as fast as the primitive operation of GPS. In other words, dovetailing a (320-bit, 80-bit) pair with respect to a 160-bit integer can be implemented at least as efficiently as multiplying a 160-bit number by an 80-bit one.

We first start with the most natural and simpler dovetailing algorithm, described in algorithm 1. This algorithm requires on average 40 additions of a 160 bit number with a 320 bit number. If we neglect the few extra additions caused by the possible propagation of the carry in some pathological cases, the number of 8-bit additions is the same as in GPS.

Now, it is almost always as fast to substract an integer as to add it. We can take advantage of this property to speed up the computation time of the algorithm by optimizing on the fly the number of bits where r and e are equal. We assume in algorithm 2 that s is equal to 1 mod 4. The case where $s = 3 \bmod 4$ is similar.

Algorithm 1 Simple dovetailing algorithm

INPUT: r, e, s
OUTPUT: $\mathcal{D}_s(r, e)$

```
i ← 0
while i < |e| do
  if e_i ≠ r_i then
    r ← r + s
  end if
  s ← s << 1
  i ← i + 1
end while
```

It is fairly straightforward to compute the cost of this algorithm: when r and e match on a bit, there is simply a shift of one bit and when they don't, there is one addition or one subtraction which makes them match on two bits and a shift of two bits.

Hence, the average number of addition/subtraction of a 320 bit number and a 160 bit number needed to complete a run of the algorithm is $\frac{|e|}{3} \approx 27$, which means that approximately 540 additions on 8 bit numbers are needed, which has to be compared with the 800 additions in case of GPS. It is certainly possible to optimize GPS in a similar way by introducing substractions in the usual shift-and-add algorithm for multiplication. However, this will involve carries that dovetailing does not need, and result in a slightly less efficient implementation.

One can further optimize the scheme at the cost of creating a table of precomputed small multiples of s. To compute a multiple of s to add in the dovetailing algorithm, the table could be indexed according to the difference between the last d bits of e and r, taken modulo 2^d. Given this approach, d might be made larger than 2. For example, a table with $2^8 = 256$ entries would permit $d = 8$ and reduce the cost to about 200 additions on 8 bit numbers. We will not pursue this matter here as it is unclear whether the corresponding memory can be available in the setting of low cost smart cards that we envision.

Hardware implementation We will now see that dovetailing can also be implemented fairly efficiently in hardware. Such an implementation would again be of a similar magnitude as the one for GPS.

We recall that the context of hardware implementation of these algorithms is simplicity and low cost, as they are reasonable candidates for performing digital signatures on memory cards. As a consequence, we considered the simplest and most compact designs.

The simplest way to implement a multiplication in hardware is by using the shift and add paradigm. Typically, one input is presented in bit parallel form while the other is in bit serial form. Each bit in the serial input "multiplies" the parallel input by either 0 or 1. The parallel input is held constant while each bit of the serial input is presented. Then, the result from each bit is added to an accumulated sum, which is shifted one bit before the next addition.

Algorithm 2 Twobits dovetailing algorithm

INPUT: r, e, s
OUTPUT: $\mathcal{D}_s(r, e)$

```
i ← 0
while i < |e| do
  if e_i = r_i then
    s ← s << 1
    i ← i + 1
  else
    if i = |e| − 1 then
      r ← r + s
      i ← i + 1
    else
      t = (2r_{i+1} + r_i − 2e_{i+1} − e_i) mod 4
      if t = 1 then
        r ← r − s
      else
        r ← r + s
      end if
      s ← s << 2
      i ← i + 2
    end if
  end if
end while
```

Now, when working on fairly large numbers, one will use a chain of unitary full-adders (say 8 bits), and might not want to wait until the carries fully propagate. This can be fairly easily solved by adding, at each cycle, the current value of the result, the value of this round input *and* the carry of the previous cycle. When all the additions have been performed the remaining carry is propagated.

At first, it seems to be harder to do such an optimization with dovetailing because the current value of the result should be used in the input test. However, we remark that in the implementation of dovetailing described above, the equality tests of the loop can be performed *before* the addition is completed. As a matter of fact, while it is necessary to eventually complete the additions so that r yields the correct result, we just need the $i-th$ bit of r to be correct when performing the $i-th$ iteration of the loop.

Hence it seems possible to implement a hardware design following the general principle described below:

- The bits of r and e are compared to figure out whether to add 0 or s to r (easily done with a `MUX`).
- Additions are made with full adders of some given small size, say 8 bits.
- Every addition between r and s is performed in parallel on all the 8 bits full adder, *but* the carries between full adders are propagated only *once*. Remaining carries are reinserted in the full adders at the next round.
- s is shifted.

Because we propagate the carry only once every round, each addition between r and s only needs 2 clock cycles, and the carry propagation only need to be performed once after the main loop.

Going into more details, we note that it is in fact necessary to perform the extra propagation step only once every 8 cycles (if we are working with 8-bit full adders). So, at the price of a more complex clocking algorithm, it might be possible to perform dovetailing in hardware at virtually the same cost as the GPS multiplication.

4 Conclusion

We have shown an attack against the OTM signature scheme [8] presented at Financial Cryptography 2002. Although the scheme can be repaired in a straightforward manner by raising the parameters, this drastically degrades its performances in terms of computing time, making it less efficient than GPS. We have also designed another scheme, based on a novel operation called dovetailing, which can be implemented in a surprisingly efficient way. This scheme appears to achieve performances similar to OTM in terms of key size, and similar to GPS in terms of computing power requirements, thus yielding a strictly better scheme than both of them, by matching their respective strengths. We argue that both our scheme and GPS plausibly qualify as a candidate for execution on a memory card rather than a smart card. Still, our scheme is better in terms of signature size.

5 Acknowledgements

We wish to thank Marc Girault, Guillaume Poupard and Adi Shamir for nice discussions around the GPS scheme and its variants. We also thank the anonymous referees for their constructive remarks and suggestions. Finally, we wish to mention the respective contributions to the present work of the two authors: the first named author is responsible for the cryptanalysis, while the second named author has found the new design and the corresponding implementations.

References

1. A. Bosselaers and R. Govaerts and J. Vandewalle Comparison of three modular reduction functions *Proceedings of Crypto 93*, Lecture Notes in Computer Science 773, pages 175-186, 1994.
2. U. Feige and A. Fiat and A. Shamir. Zero-Knowledge Proofs of Identity. *J. Cryptology*, 1, 1988, 77–94.
3. S. Even, O. Goldreich and S. Micali. Online/Offline Digital Signatures. In *Journal of Cryptology*, 9, 1996, 35–67.
4. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Proceedings of Crypto 86*, Lecture Notes in Computer Science 263, 1987, 181–187.

5. M. Girault. Self-certified Public Keys. In *Proceedings of Crypto 91*, Lecture Notes in Computer Science 547, Springer-Verlag, 1992, 490–497.
6. L.S. Guillou and J.-J. Quisquater. A practical zero-knowledge protocol fitted to security microprocessors minimizing both transmission and memory. In *Proceedings of Eurocrypt 88*, Lecture Notes in Computer Science 339, Springer-Verlag, 1989, 123–128.
7. N. Lygeros, M. Mizony, P. Zimmermann, A new ECM record with 54 digits, `http://www.desargues.univ-lyon1.fr/home/lygeros/Mensa/ecm54.html`
8. T. Okamoto, M. Tada and A. Miyaji. An Improved Fast Signature Scheme without on-line Multiplication. In *Pre-proceedings of Financial Cryptography 02*.
9. J. Pollard. Monte Carlo methods for index computation mod p, *Math. Comp.*, 32, 1978, 918–924.
10. G. Poupard and J. Stern. Security Analysis of a Practical "on the fly" Authentication and Signature Generation. In *Proceedings of Eurocrypt 96*, Lecture Notes in Computer Science 1403, Springer-Verlag, 1998, 422–436.
11. G. Poupard and J. Stern. On the fly Signatures based on Factoring. In *Proceedings of the 6th ACM Conference on Computer and Communications Security*, ACM Press, 1999, 48–57.
12. C. P. Schnorr. Efficient Identification and Signatures for Smart Cards. In *Proceedings of Crypto '89*, Lecture Notes in Computer Science 435, Springer-Verlag, 1990, 235–251,
13. C. P. Schnorr. Efficient Signature Generation by Smart Cards. *Journal of Cryptology*, 4, 1991, 161–174.
14. A. Shamir and Y. Tauman. Improved Online/Offline Signatures Schemes. In Crypto 2001, Lecture Notes in Computer Science 2139, Springer-Verlag 2001, 355–367.