On the Security of Some Variants of the RSA Signature Scheme

Markus Michels[1], Markus Stadler[1], and Hung-Min Sun[2]

[1] Ubilab, UBS
Bahnhofstr. 45
8021 Zurich, Switzerland
{Markus.Michels,Markus.Stadler}@ubs.com

[2] Department of Information Management
ChaoYang University of Technology
Wufeng, Taichung County
Taiwan 413
hmsun@dscs.csie.nctu.edu.tw

Abstract. We describe adaptive attacks on several variants of the RSA signature scheme by de Jonge and Chaum. Moreover, we show how to break Boyd's scheme with an adaptive, a directed and a known signature attack. The feasibility of the adaptive attack on Boyd's scheme is illustrated by a concrete example.

Keywords: cryptanalysis, digital signature schemes, RSA variants.

1 Introduction

Since the invention of the RSA signature scheme [20] in 1978, many variants have been proposed in order to improve the security and the efficiency of the scheme. However, these attempts have not always been successful. For instance, Rabin [19] and Williams [22] suggested schemes whose security can be proved equivalent to factoring the modulus as long as the attacker is assumed to be passive, but which are vulnerable to so-called adaptive attacks (see [11] for a formal description of attacker models on signature schemes). Proposals to increase the efficiency, such as to use the same modulus for all users, have shown to be vulnerable ([8, 12] respectively) as well. Finally, it was suggested to use other group structures, such as elliptic curves or Lucas functions [14, 9, 17], but as was discovered recently, these variants do not provide better security than the basic scheme [1–3].

In 1986 de Jonge and Chaum suggested a generalization of the RSA signature scheme [13] and discussed several of its instances. In particular, they showed that many special cases are vulnerable to adaptive attacks [7] that make use of the multiplicative property of the RSA scheme. They came up with a specific variant they claimed to be resistant against such attacks. Recently, Boyd proposed another special case of this generalized scheme [4] that works in a relatively small subgroup of $\mathbb{Z}_n^*$ and is therefore quite efficient.

Our Contribution:
In this paper we will first point out that de Jonge–Chaum signature scheme is vulnerable to adaptive attacks as well. We then describe in detail an adaptive attack on Boyd's scheme and give an analysis of its runtime and its success probability. For a concrete realization of the scheme, with RIPEMD-160 [10] used as hash-function, we show a successful attack: if the attacker manages to obtain from a user the signatures on the following six messages (for instance as payment for an electronic service),

"Check #00000000167658: value $1.00",
"Check #00000000347533: value $1.00",
"Check #00000003719673: value $1.00",
"Check #00000007303360: value $1.00",
"Check #00000014393409: value $1.00",
"Check #00002271656103: value $1.00",

he can also compute a valid signature (with respect to this user's public key) on the message

"Check #00002228615476: value $1'000'000.00" .

This attack works because the 160 bit hash-values produced by RIPEMD-160 are quite likely to be the product of relatively short factors (in this example less than 32 bits). This attack is then extended to a directed and a known-signature attack. Finally we outline how these ideas could be used to attack a naive implementation of the RSA blind signature scheme.

Organization of the Paper:
In Section 2 we review the de Jonge–Chaum scheme and describe our attack. In Section 3, we focus on Boyd's scheme, describe our cryptanalysis and the concrete attack. Section 4 concludes the paper with a discussion of the potential vulnerability of a naive implementation of RSA blind signature scheme.

2 An Adaptive Attack on the de Jonge–Chaum Scheme

2.1 Review of the de Jonge–Chaum Scheme

In [13], de Jonge and Chaum consider several variations of the basic RSA signature scheme with the goal to avoid adaptive attacks. These variations can be characterized by the verification equation

$$\sigma^{f_1(m,n)} \equiv f_2(m,n) \pmod{n} ,$$

where n is the signer's public modulus, m is the message, σ is the signature, and $f_1, f_2 : \{0,1\}^* \times \mathbb{Z} \to \mathbb{Z}$ are efficiently computable functions mapping (m,n) to an integer. For example, the choices

$$f_1(m,n) = e \quad \text{and} \quad f_2(m,n) = m$$

for an integer constant e with $\gcd(e, \varphi(n)) = 1$ result in the standard RSA signature scheme (without hashing).

Since the signer knows the factorization of n he can compute the $f_1(m,n)$-th root of $f_2(m,n)$ (if it exists) which is signature σ on m.

For several choices of f_1 and f_2 de Jonge and Chaum show that adaptive attacks on the resulting signature scheme are possible and finally come up with a scheme they think is secure in this respect, i.e. the variant with the verification equation

$$\sigma^{2m+1} \equiv m \pmod{n} .$$

They claim explicitly that this scheme is secure for messages of arbitrary length and that messages need not contain redundancy. We will show in the next subsection that under these assumptions multiplicative attacks are possible.

2.2 The Adaptive Attack

An adaptive attack on the above scheme works as follows. Assume the attacker wants to construct a signature on a message m by having the signer sign two other messages m_1 ($\neq m$) and m_2 ($\neq m$). He constructs m_1 and m_2 as follows.

- choose an arbitrary integer $u > 0$ and compute

$$m_1 := \frac{(2m+1)\cdot(2u+1)-1}{2}$$

- compute integers k and j satisfying the equation

$$2mm_1 + 2kn + 1 = j \cdot (2m+1) .$$

  This can be done by first computing

$$k \equiv \frac{-2mm_1 - 1}{2n} \equiv \frac{m_1 - 1}{2n} \pmod{2m+1}$$

  (if $\gcd(n, 2m+1) = 1$) and then computing j accordingly.

- finally compute $m_2 := mm_1 + kn$.

The attacker then has the signer to issue signatures σ_1 and σ_2 on the messages m_1 and m_2, respectively. With the values u, k, and j he is now able to construct a signature on m by computing

$$\sigma := \frac{\sigma_2^j}{\sigma_1^{2u+1}} .$$

This is a valid signature on m since

$$\sigma^{2m+1} \equiv \frac{\sigma_2^{j\cdot(2m+1)}}{\sigma_1^{(2u+1)\cdot(2m+1)}} \equiv \frac{\sigma_2^{2mm_1+2kn+1}}{\sigma_1^{2m_1+1}} \equiv \frac{\sigma_2^{2m_2+1}}{m_1}$$
$$\equiv \frac{m_2}{m_1} \equiv m \pmod{n} .$$

Note that this attack does not work anymore if messages must have redundancy or if the length of messages is limited.

2.3 An Attack on the Generalized Scheme

We can adapt the attack to the general scheme with verification equation $\sigma^{f_1(m,n)} \equiv f_2(m,n) \pmod{n}$ if f_1, f_2 satisfy certain conditions, e.g. an efficient function $\bar{f}_2$ must exist such that $f_2(\bar{f}_2(m,n),n) = m$. Here the attacker constructs m_1 and m_2 as follows.

- He finds integers u, m_1 with $u \neq 0$ such that
$$f_1(m_1,n) := f_1(m,n)u,$$
- computes integers k and j satisfying the equation
$$f_1(\bar{f}_2(f_2(m,n)f_2(m_1,n)+kn,n),n) = j \cdot f_1(m,n) \text{ and}$$
- finally computes $m_2 := \bar{f}_2(f_2(m,n)f_2(m_1,n)+kn,n)$.

As before, the attacker then has the signer to issue signatures σ_1 and σ_2 on the messages m_1 and m_2, respectively. Hence a signature on m can be calculated by $\sigma := \frac{\sigma_2^j}{\sigma_1^u}$. This is a valid signature on m since

$$\sigma^{f_1(m,n)} \equiv \frac{\sigma_2^{j\cdot f_1(m,n)}}{\sigma_1^{u\cdot f_1(m,n)}} \equiv \frac{\sigma_2^{f_1(\bar{f}_2(f_2(m,n)f_2(m_1,n)+kn,n),n)}}{\sigma_1^{f_1(m_1,n)}} \equiv \frac{\sigma_2^{f_1(m_2,n)}}{f_2(m_1,n)}$$
$$\equiv \frac{f_2(m_2,n)}{f_2(m_1,n)} \equiv f_2(m,n) \pmod{n}.$$

3 Attacking Boyd's Scheme

In this section we will focus on a signature scheme which was recently suggested by Boyd [4].

3.1 Review of Boyd's Scheme

Boyd's scheme works a follows:

- *Key generation:* Each user chooses two large primes p and q such that both $p-1$ and $q-1$ have a large prime factor and then computes $n := pq$. He further chooses a prime $r \approx 2^{160}$ with $r|(p-1)$ and a generator g of order r in $\mathbb{Z}_n^*$. He keeps p, q,and r secret and publishes g and n. Moreover, a collision resistant hash function h is chosen.
  Remark: It was pointed out by Meijer [4] that n can be factored if $r \not| (q-1)$, as $g \pmod{q} = 1$ and thus $\gcd(g-1,n) = q$. Moreover, it was shown by Mao [16] that $n-1$ is a multiple of r if r divides both $p-1$ and $q-1$ and therefore computing RSA-roots in the subgroup of order r is easy. We assume in the following that r is chosen such that given the public parameters it is infeasible to compute a multiple of r. Hence r must be composite and of suitable size.

- *Signature generation:* To sign a message m (with $h(m) \not\equiv 0 \pmod{r}$)) the signer computes $d := h(m)^{-1} \pmod{r}$ and $s := g^d \pmod{n}$. Then s is a signature on the message m.
- *Signature verification:* A signature s on m can be verified by checking whether $s^{h(m)} \equiv g \pmod{n}$ holds.

It is suggested to use a 1024-bit modulus n and a hash function h that outputs 160-bit values. With these choices of the security the efficiency of the scheme compares favorably with the RSA signature scheme or the DSS scheme.

3.2 Basic Observations

In the above scheme, signatures are just different roots of g, where the message determines which root it is. Our adaptive attack will make use of the following simple facts:

1. If s is an $(a \cdot b)$-th root of g, i.e. $s^{ab} \equiv g \pmod{n}$, then $s^a \pmod{n}$ and $s^b \pmod{n}$ are b-th and a-th root of g, respectively (this was already observed in [13]).
2. If s_a and s_b are a-th and b-th roots of g, respectively, and if $\gcd(a, b) = 1$, then it is possible to find an $(a \cdot b)$-th root of g by first computing integers u and v satisfying $ua + vb = 1$ (using extended Euclid's algorithm) and then computing $s_{ab} = s_a^v s_b^u \pmod{n}$ as $s_{ab}^{ab} \equiv s_a^{abv} s_b^{abu} \equiv g^{bv+au} \equiv g \pmod{n}$.

3.3 An Adaptive Attack

The goal of the adaptive attack is to forge a signature on a message m by having the signer issue signatures on some messages $m_i \neq m$ for $i = 1, \ldots, k$ chosen by the attacker. The idea is to choose m such that

$$h(m) = g_1 \cdot g_2 \cdot \ldots \cdot g_k$$

where $g_1, \ldots, g_k$ are relatively small integers that are pair-wise coprime. Then k messages $m_1, \ldots, m_k$ are chosen such that g_i divides $h(m_i)$ for $i = 1, \ldots, k$. Using the signatures on the messages $m_1, \ldots, m_k$ the attacker can derive g_1-th, $\ldots$, g_k-th roots of g (fact 1) and then compute the $(g_1 \cdot g_2 \cdot \ldots \cdot g_k)$-th root of g (fact 2) which is the signature on m. More precisely, the adaptive attack works as follows.

- *Determining m:*
  In order to forge a signature on a message m', the attacker calculates a similar message m (i.e. the semantic content of m and m' is the same) such that $h(m) = \prod_{i=1}^{k} g_i$ for pair-wise coprime numbers $g_i \leq 2^b$ for $1 \leq i \leq k$.
- *Finding $m_1, \ldots, m_k$:*
  Let $L := \{\}$ and $i = 1$. Repeat the following step unless $i = k + 1$: The attacker picks a random message m_i and computes $h(m_i)$. If $gcd(h(m_i), g_i) \neq 1$, he computes $L := L \cup \{m_i\}$ and $i := i + 1$.

– *Obtaining signatures:*
  Ask the signer for signatures $s_1, \ldots, s_k$ to all messages in $L = \{m_1, \ldots, m_k\}$ and combine these signatures using facts 1 and 2 to obtain a signature s on $h(m)$.

We have to estimate the probability that an arbitrary $h(m)$ consists of pair-wise coprime $(b-1)$-bit factors only and to determine c such that the probability that each of these factors is a factor in one of the hash values $h(m_1), ..., h(m_c)$ for arbitrary messages $m_1, \ldots, m_c$ is close to 1.

Smoothness of a Hash Value

We can try to find a lower bound for the probability that an arbitrary output of the hash function h can be fully factored into pair-wise coprime factors smaller than 2^b (provided the hash function h is a truly random function). Let $|h(x)| = lb$ for a certain $l \geq 2$ and arbitrary messages x. Some "good cases" can be described as follows:

1. $h(m) = z \prod_{i=1}^{l-1} p_i$, where $0 < z < 2^{b-t}$, the p_i's are prime and pairwise distinct, $2^{b-t} < p_i < 2^b$ for $1 \leq i \leq l-1$ and $0 < t < b$.
2. $h(m) = \prod_{i=1}^{l} p_i$, where the p_i's are prime and pairwise distinct, $2^{b-t} < p_i < 2^b$ for $1 \leq i \leq l$.
3. For $2 \leq j \leq l-1$: $h(m) = \prod_{i=1}^{l} p_i$, where the p_i's are prime and pairwise distinct, $0 < p_i < 2^{b-t}$ for $1 \leq i \leq j$ and $2^{b-t} < p_i < 2^b$ for $j+1 \leq i \leq l$.

With $\pi(2^b) \approx \frac{2^b}{b\ln(2)} \approx \frac{2^b}{0.7b}$ and $\delta = \frac{(b-t)2^b - b2^{b-t}}{0.7b(b-t)}$ an approximation for the number of the above listed good cases ψ is

$$\psi \approx \frac{2^{b-t}\delta^{l-1}}{(l-1)!} + \frac{\delta^l}{l!} + \sum_{j=2}^{l-1} \frac{\left(\frac{2^{b-t}}{0.7(b-t)}\right)^j \delta^{l-j}}{(l-j)!j!}$$

For concrete values b, l and $t = 3$ (to get an optimal result) the following lower bounds for the probability $Pr(= \psi/2^{bl})$ that an arbitrary $h(m)$ has only pair-wise coprime factors smaller than 2^b can be derived.

b	l	$\lvert h(m) \rvert$	$-log_2(Pr)$
20	8	160	42.4
23	7	161	36.8
27	6	162	31.1
32	5	160	25.3
40	4	160	19.6
30	8	240	46.5
40	6	240	33.9
32	10	320	62.7
40	8	320	49.4

Finding Signatures with Suitable Factors

We would like to find a value c that the probability that each element of $L = \{g_1, \dots, g_k\}$ is a factor of one element in $M = \{h(m_1), \dots, h(m_c)\}$ is close to 1, i.e. find c and

$$\rho := Pr(\exists (j_1, \dots, j_k) : \forall i : g_i | h(m_{j_i}) || 1 \leq i \leq k, 1 \leq j_i \leq c)$$

such that $\rho \approx 1$. We define

$$\rho' := Pr(\exists (j_1, \dots, j_k) : \forall i : g_i | h(m_{j_i}) ||$$
$$1 \leq i \leq k, 1 \leq j_i \leq c, j_a \neq j_v, 1 \leq a < v \leq k)$$

and obviously $\rho \geq \rho'$. For a given $g_j \in L$

$$Pr(\exists i : g_j | h(m_i) || 1 \leq i \leq c) \geq 1 - \left(1 - 2^{-b}\right)^c$$

holds. Let us focus on the following experiment:

- $M_1 = M$.
- For $i = 2, \dots, k$: $M_i = M_{i-1} - \{d\}$, where $d \in M_{i-1}$ and $g_{i-1} | d$ if g_{i-1} divides at least one element in M_{i-1}. Otherwise d is uniformly chosen in M_{i-1}.
- For $i = 1, \dots, k$: The event X_i is that g_i divides at least one element in M_i.

The probability that all $X_1, ..., X_k$ hold is a lower bound for ρ'. Let us show that $Pr(X_1, \dots, X_k) > Pr(X_1) \cdots Pr(X_k)$ holds. We try to find a lower bound for the probability $Pr(X_2 | X_1)$. Assume that g_1 does not divide any element in M_1. Hence the probability that g_2 divides at least one of the remaining $c-1$ elements is higher than if these elements were chosen at random. Assume that g_1 divides at least one element in M_1. The probability that g_1 divides at least one element in M_2 is slightly smaller than if these elements were chosen at random. Hence the probability that g_2 divides at least one element in M_2 is slightly higher than if these elements were chosen at random. Hence $Pr(X_2 | X_1) > Pr(X_2)$ and in a similar way $Pr(X_j | X_1, \dots, X_{j-1}) > Pr(X_j)$ for $j = 3, \dots, k$ can be shown. As a result we have

$$\rho' > \prod_{i=1}^{k} 1 - \left(1 - 2^{-b}\right)^{c+1-i} > 1 - \sum_{i=1}^{k} \left(1 - 2^{-b}\right)^{c+1-i} =$$

$$1 - \left(1 - 2^{-b}\right)^{c+1} \sum_{i=1}^{k} \left(\frac{2^b}{2^b - 1}\right)^i = 1 - \left(1 - 2^{-b}\right)^{c+1} \frac{\left(\frac{2^b}{2^b-1}\right)^{k+1} - \left(\frac{2^b}{2^b-1}\right)}{\frac{2^b}{2^b-1} - 1} =$$

$$1 - \left(1 - 2^{-b}\right)^{c+1} \left(\frac{2^{b(k+1)}}{(2^b - 1)^k} - 2^b\right) = 1 - 2^b (1 - 2^{-b})^{c+1} \left((1 - 2^{-b})^{-k} - 1\right)$$

We get the following values for c such that $\rho > \rho' > 0.99$ using the above derived lower bound for ρ'. Note that $l \leq k \leq bl$.

b	k	$\log_2 c$
20	8	23
20	16	23
27	6	30
27	12	30
32	5	35
32	10	35
40	4	43
40	8	43

As a result we can conclude that the choice of $|h(m)| = 160$ cannot be considered as secure.

Practical Results

As was already mentioned in the introduction, we have found a concrete adaptive attack for a scheme using the RIPEMD-160 hash-function [10]. RIPEMD outputs 160 bit strings which were directly interpreted as integers (where the leftmost bit is interpreted as the most significant bit). The attack was implemented in C using the LIP long integer package [15] and executed on a SUN Enterprise Server. First, we tried to find a "target" message m of the form

"Check #xxxxxxxxxxxxxx: value $1'000'000.00"

where "xxxxxxxxxxxxxx" is a 14-digit decimal integer that served as a counter. Since factoring the resulting hash-value is quite time consuming, only those hash-values were factored whose most significant 20 bits were all zero and which therefore are more likely to contain only short factors. Factoring was done in two steps, first trial division with all prime factors smaller than 2^{20} and then Pollard's rho method [18] with 2^{15} steps in the main loop (using LIP's `zpollardrho` function [15]) to find the remaining (small) prime factors. On a SUN Enterprise Server this program did run for approximately one day to find the following result.

$$\begin{aligned}&\text{RIPEMD}_{160}(\text{“Check \#00002228615476: value \$1'000'000.00”}) = \\ &= 591958810961311141109102519582266871126124 \\ &= 230589 \cdot 581003 \cdot 5743124 \cdot 30989939 \cdot 44307239 \cdot 560313293\,.\end{aligned}$$

Note that the six factors are pair-wise coprime but not necessarily prime. In the next step we were searching six messages of the form

"Check #xxxxxxxxxxxxxx: value $1.00"

whose RIPEMD-160 hash value is divisible by the six factors of the target message. Using simple trial division, this took us less than a day on the SUN Enterprise Server. The six messages are:

$$\begin{aligned} &\mathrm{RIPEMD}_{160}(\text{“Check \#00000000167658: value \$1.00”}) = \\ &\quad = 230589 \cdot 31421675957870400101948568611014772573448116 \end{aligned}$$

$$\begin{aligned} &\mathrm{RIPEMD}_{160}(\text{“Check \#00000000347533: value \$1.00”}) = \\ &\quad = 581003 \cdot 4928106028851136548536883809702992290842257 \end{aligned}$$

$$\begin{aligned} &\mathrm{RIPEMD}_{160}(\text{“Check \#00000003719673: value \$1.00”}) = \\ &\quad = 5743124 \cdot 93987921877576599808444485864739254202422 \end{aligned}$$

$$\begin{aligned} &\mathrm{RIPEMD}_{160}(\text{“Check \#00000007303360: value \$1.00”}) = \\ &\quad = 44307239 \cdot 2858765935326482727398427810809271406 1121 \end{aligned}$$

$$\begin{aligned} &\mathrm{RIPEMD}_{160}(\text{“Check \#00000014393409: value \$1.00”}) = \\ &\quad = 30989939 \cdot 21093763551230032693705837318819942077090 \end{aligned}$$

$$\begin{aligned} &\mathrm{RIPEMD}_{160}(\text{“Check \#00002271656103: value \$1.00”}) = \\ &\quad = 560313293 \cdot 23146180939659603936317453322531208165552 \end{aligned}$$

3.4 A Directed Attack

We can slightly modify the adaptive attack in order to get an attack for the weaker model of a directed attack. Here the attacker is just allowed to get signatures on all messages in a set A chosen by himself once. The aim of the attack is to find a signature to messages which are not in A.

- The attacker chooses a basis B that contains the maximal prime powers that are smaller than a certain number, say 2^b. We denote $B = \{g_1, \cdots, g_k\}$, where $g_i = p_i^{e_i}$, $e_i \in \mathbb{Z}_{>0}$, p_i is prime and $p_i^{e_i} \leq 2^b \leq p_i^{e_i+1}$ for all $i = 1, \ldots, k$ and the p_i's are pairwise distinct. Let $D = B' = \{\}$.
- The attacker repeats the following step until $B = B'$: He picks a new random d, computes $h(d)$ and finds all factors of $h(d)$ that are in B. Thus $h(d) = z \cdot \prod_{i=1}^{k} g_i^{w_i}$, where z has no factors that are in B and $w_i \in \{0, 1\}$. If there is a prime power factor g_i in $h(d)$ with $g_i \notin B'$, the attacker adds (d, g_i) to the set D and g_i to the set B'.
- The attacker asks the signer to sign the message d for all pairs (d, g_i) in D. Using fact 1 mentioned in subsection 3.2 he can get parameters s_i to each g_i such that $s_i^{g_i} = g \pmod{n}$ for all $i = 1, \ldots, k$.
- In order to forge a signature on a message m', the attacker calculates a similar message m (where the semantic content of m and m' is the same) such that $h(m) = \prod_{i=1}^{k} (p_i^{f_i})^{w_i}$ where $f_i \in \mathbb{Z}_{>0}$, $0 < f_i \leq e_i$ and $w_i \in \{0, 1\}$ for all $i = 1, \ldots, k$.

- Then using the known signatures to all elements in B and by applying the facts 1 and 2 mentioned in subsection 3.2, the attacker can compute s such that $s^{h(m)} = g \pmod{n}$.

Analysis

We can apply the analysis of the adaptive attack. The probability that an arbitrary hash value $h(m)$ has only pair-wise coprime prime power factors smaller than 2^b with given b is the same as calculated above. We can also modify the analysis of finding signatures with suitable factors. As $k \approx \frac{2^b}{b \ln 2} \approx \frac{1.42 \cdot 2^b}{b}$ now, we get the following values for the number of hash values c such that $\rho > \rho' > 0.99$ using the lower bound for ρ'.

b	$\log_2 c$
20	24
27	32
32	37
40	45

Hence we can conclude again that the choice of $|h(m)| = 160$ is to small to make this attack infeasible. It should be noted that the number of signatures needed for this attack is at most the size of the basis $k \approx \frac{1.42 \cdot 2^b}{b}$.

3.5 Known-Signature Attack

In an even weaker attack model, the known-signature attack, the attacker may only collect signature-message pairs from its victim. The attack described above can be modified to work in this model as well with overwhelming probability, if the attacker collects about c ($\approx 2^{b+5}$ according to the table above) signature-message pairs provided they are uniformly chosen. However, the computational effort compared to the directed attack increases.

4 Remarks on a Naive Implementation of the RSA Blind Signature Scheme

Let us finally by point out, how the attack described in the last section can be adapted to a naive implementation of the RSA blind signature scheme [6]. We assume that this attack is generally known, but to the best of the authors knowledge there is no description of it in the literature.

Let the verification equation of the RSA implementation be given by

$$\sigma^e \equiv h(m) \pmod{n},$$

where n is the modulus, e is the public exponent, σ is the signature, h is a hash function that outputs at most b-bit values and b is relatively small, say $b = 160$. Then a similar attack as described in the section 3.4 works as follows: The attacker chooses a basis B that contains the all primes that are smaller than a certain number, say 2^b. We denote $B = \{p_1, \cdots, p_k\}$. Then he asks the signer to sign all $p_1, \ldots, p_k$ using the blind signature scheme (note that the signer cannot learn the unblinded messages and hence cannot check that these messages are not hash values) and gets the signatures $\sigma_1, \ldots, \sigma_k$. Now the attacker can forge signatures on messages $h(m)$ that can be totally factored over B, i.e. $h(m) = \prod_{i=1}^{k} p_i^{w_i}$ for integers $w_i \geq 0, 1 \leq i \leq k$. He can easily compute the signature σ on m by $\sigma := \prod_{i=1}^{k} \sigma_i^{w_i} \pmod{n}$.

The attack can be easily prevented by using an increased value b or by suitable padding, e.g. $m' := a_1 || \ldots || a_t$ with $a_1 := m$ and $a_i := h(a_1 || \ldots || a_{i-1})$ for $1 < i \leq t$ and $|m'| = |n|$ instead of $h(m)$. This method of padding is used in the RSA based ecash^{TM} system [21], possibly to countermeasure the above mentioned attack as well.

5 Acknowledgments

The first two authors were supported by the Swiss National Foundation (SNF/SPP project no. 5003-045293).

References

1. D. Bleichenbacher, "On the Security of the KMOV Public Key cryptosystem", LNCS 1294, Proc. Crypto'97, Springer-Verlag, (1997), pp. 235–248.
2. D. Bleichenbacher, W. Bosma, A. Lenstra, "Some remarks on Lucas-based cryptosystems", LNCS 963, Proc. Crypto'95, Springer-Verlag, (1997), pp. 386–396.
3. D. Bleichenbacher, M. Joye, J.-J. Quisquater, "A new and optimal chosen-message attack on RSA-type cryptosystems", LNCS 1334, Proc. Information and Communications Security - ICICS'97, Springer-Verlag, (1997), pp. 302–313.
4. C. Boyd, "Digital Signature and Public Key Cryptosystem in a Prime Order Subgroup of $\mathbb{Z}_n^*$", LNCS 1334, Proc. ICICS'97, Springer-Verlag, (1997).
5. Communicated by Colin Boyd.
6. D. Chaum, "Security without identification: transaction systems to make big brother obsolete", Communications of the ACM, 28, 10, October, (1985), pp. 1030–1044.
7. G.I. Davida, "Chosen signature cryptoanalysis on the RSA (MIT) public key cryptosystem", Tech. Rep. TR-CS-82-2, Department of Electrical Engineering and Computer Science, University of Wisconsin, Milwaukee, (1982).
8. J.M. DeLaurentis, "A further weakness in the common modulus protocol for the RSA cryptoalgorithm", Cryptologia, No.8, Vol. 3, (1984), pp. 253–259.
9. N. Demytko, "A new elliptic curve based analogue of RSA", LNCS 768, Proc. Eurocrypt'93, Springer Verlag, (1994), pp. 40–49.

10. H. Dobbertin, A. Bosselaers, B.Preneel, "RIPEMD-160, a strengthened version of RIPEMD", LNCS 1039, Proc. Fast Software Encryption, Springer-Verlag, (1996), pp. 71-82.
11. S. Goldwasser, S. Micali, R. Rivest, "A digital signature scheme secure against adaptive chosen message attacks", SIAM Journal on Computing, Vol. 17, No. 2, (1988), pp. 281 - 308.
12. J. Hastad, "On using RSA with low exponent in a public key network", LNCS 218, Proc. Crypto '85, Springer Verlag, (1986), pp. 404–408.
13. W. de Jonge, D. Chaum, "Some Variations on RSA Signatures & their Security", LNCS 263, Proc. Crypto '86, Springer Verlag, (1987), pp. 49–59.
14. K. Kurusawa, U. Maurer, T.Okamoto, S.Vanstone, "New public key schemes based on the ring $\mathbb{Z}_n$", LNCS 576, Proc. Crypto'91, Springer Verlag, (1992).
15. A. Lenstra, P. Leyland: LIP - Long Integer Package, 1995.
16. W.Mao, "Cryptanalysis in Prime Order Subgroups in $\mathbb{Z}_n^*$", manuscript, (1998), available at http://grouper.ieee.org/groups/1363/contrib.html.
17. W.B. Müller, W. Nöbauer, "Some remarks on public key cryptosystems", Studia Sci. Math. Hung., Vol. 16, (1981), pp. 71–76.
18. J. Pollard, "A Monte Carlo Method for Factorization", BIT, Vol. 15, (1975), pp. 331–334.
19. M.O. Rabin, "Digitalized signatures and public-key functions as intractable as factorization", MIT/LCS/TR-212, MIT Lab. for Computer Science, Cambridge, Mass., (1979).
20. R.L. Rivest, A. Shamir, L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems", Communications of the ACM, Vol.21, (1978), pp. 120–126.
21. B. Schoenmakers, "Basic Security of the ecash^{TM} Payment System", LNCS, Proc. Computer Security and Industrial Cryptography: State of the Art and Evolution, Springer-Verlag, (1997).
22. H.C. Williams, "A modification of the RSA Public-Key Cryptosystem", IEEE Trans. in Inform. Theory, IT-26, No. 6, (1980), pp. 726–729.