

# On Server-Aided Computation for RSA Protocols with Private Key Splitting

Anne-Maria Ernvall<sup>1\*</sup> and Kaisa Nyberg<sup>2</sup>

<sup>1</sup> Department of Mathematics, University of Helsinki, Finland

<sup>2</sup> Nokia Research Center, Helsinki, Finland

September 23, 2003

## Abstract

Server-aided secret computation is a widely investigated topic due to the obvious benefits of such an approach in applications where small constrained terminal equipment is used. Recently, also other ways of using servers to improve the security functionality of end user devices have been proposed. By splitting the private computation functionality between the server and the client, end user devices can be protected from being misused if captured, or the usage of the private key can be controlled. In this contribution the problem of combining different security tasks delegated to the servers by client devices is investigated. The capture resilience and delegated authorisation protocols based on RSA is achieved by composition of the private key as a sum of two partial keys. To reduce the computation of one party, its partial key must be reduced. Previously, continuous fractions were used to break RSA with a short private key. We apply this method to the case where the private RSA key itself is large but it is split into two parts, one of which is small. Our results show that for some typical parameter values the party who knows the longer part of the private key can easily recover the small part.

**Key words:** E- and M-business security, server-aided secure computation, capture-resilience, delegated authorisation, RSA system, small RSA exponents

## 1 Introduction

Server-aided computation protocols are protocols between two parties, the client and the server. Cryptographic computations, particularly those related to some public key system, often involve heavy, time and power consuming operations. The problem is how the client can outsource computations that use some private information known to the client, without revealing the private information to the aiding server. Typical applications of server-aided protocols include smart card computations, which may need to be aided using an external processor within the host device. The computations required to be performed by mobile devices often need to be reduced to the minimum because of constrained electric power resources.

---

\*The author's work on this paper was performed during her internship at Nokia Research Center

Numerous protocols for providing server-aided functionality for the common public key systems have been proposed and analysed. For a review of the most important developments in this area see [5] where also a new protocol for server-aided RSA is proposed. Other previous literature highlighting the possible pitfalls include [3] and [11]. For the purposes of this paper we are interested in server-aided RSA protocols, and in particular those not using the Chinese Remainder Theorem.

The goal of this work is to investigate if it is possible to include the server-aided facility to certain new protocols using RSA, which share the private key functionality between a client and a server to provide other security services. In [8] and [9] P. MacKenzie and M. Reiter presented protocols where the private RSA key is split into two parts and given to two parties in such a manner that neither party cannot perform the private RSA operation alone. One of the parties, often an end user device in practise, then produces the result of the private RSA operation using the assistance of the second party. Subsequently, S. Sovio, N. Asokan and K. Nyberg observed that this idea also applies to the case where an owner of a private RSA key wants to delegate the capability of using the RSA key to other devices, while retaining some control of how the key is used by those devices [13]. For example, the owner might want to set a limit to the amount of financial transaction that can be signed using the key. In the Sovio-Asokan-Nyberg protocol the control function is delegated to an assistant, which is a server that is available online, by giving it a share of the private RSA key. Also further delegation is possible using the help of the same assistant.

The main mathematical tool used in this work is the continuous fraction expansion for rational numbers. Recently, this method was used by M. Wiener in [15] to show that the private RSA exponent shall not be too small. Using RSA exponent smaller than  $\frac{1}{3}\sqrt[4]{n}$ , where  $n$  is the RSA modulus, is not secure. For a presentation of this attack we refer to [1] or [14]. Later D. Boneh and G. Durfee increased this bound to  $n^{0.292}$  in [2]. They used a method based on the LLL-algorithm [7]. Recently, the feasibility of these attacks was investigated for the multiprime RSA system in [4].

In [8] MacKenzie and Reiter mention the problem of adding server-aided facility to their protocol, but leave it as a problem for future work. In the context of [8] and [13] the client and server roles can also appear to be switched, as it may be desired that the server's workload is reduced. In this paper, we study this question and give some negative results. We use continuous fraction expansions to show that adding server-aided computation facility to the protocols of MacKenzie and Reiter and Sovio et al is not always possible by making one share of the RSA key small. We show that this negative result holds if the public key is small and also if it is sufficiently large. Finally, we also consider the case where the private RSA key is decomposed as a product of two shares, one of which is small. This scenario yields completely under the continuous fractions attack.

The paper is structured as follows. First, the basic server-aided RSA protocol is given. Then the principles of the RSA splitting in the MacKenzie-Reiter and Sovio-Asokan-Nyberg protocols are described. The continued fraction expansion is described in Section 4, where we also show how it was applied to the case of short RSA private key by Wiener. Our results on the insecurity of small RSA private key shares are given in Section 5.

## 2 Server-Aided RSA Computation

The parameters of the RSA system are two large and different prime numbers  $p$  and  $q$ , their product  $n = pq$  called the RSA modulus, the private exponent  $d$  and the public exponent  $e$ . The knowledge of the private key is essentially equivalent to the knowledge of the factors  $p$  and  $q$ . For a probabilistic algorithm for deriving the factors from the knowledge of  $e$ ,  $d$  and  $n$ , see, for example, [14].

If the factors  $p$  and  $q$  are known the RSA computations modulo  $n$  can be speeded up using the Chinese Remainder Theorem. However, in the protocols discussed in this paper the factorisation is not known to the parties of the protocols. Therefore, our interest is in server-aided computation protocols that do not assume the use of the Chinese Remainder Theorem. The choice is not large. Essentially all protocols are variations of the basic protocol described below. The most recent protocol of Hong et al [5] uses a different approach, but requires that the client knows the factorisation of  $n$ .

The basic approach to server-aided RSA computations [10] is to decompose the private  $d$  as follows:

$$d = \sum_{i=1}^{\ell} w_i f_i \pmod{\varphi(n)}.$$

Creating such a decomposition requires the knowledge of  $\varphi(n)$  unless in each term one of the factors  $w_i$  or  $f_i$  is equal to 1 or 0.

Client gives the values  $w_1, w_2, \dots, w_\ell$  to the server. The effect of server-aided computation is achieved by selecting all values  $f_i$  small. To create a result of a private RSA operation on some message  $m$ , the client sends the values  $w_1, w_2, \dots, w_\ell$  and  $m$  to the server. Server computes  $\ell$  values  $m^{w_i}$ ,  $i = 1, 2, \dots, \ell$ , and returns them to the client. It then computes:

$$S = \prod_{i=1}^{\ell} (m^{w_i})^{f_i}.$$

As shown by Burns and Mitchell [3] it is essential that the client verifies the result  $S$  before publishing it. The server may try to find the values of  $f_i$  one by one using the following procedure.

When given a message  $m$  the server gives the correct values for all but one of  $m^{w_i}$ . For one of the indices, say  $j$ , the server replaces the value  $m^{w_j}$  by  $tm^{w_j}$ . If client doesn't verify the result  $S$  of the private key operation, it prompts out

$$S = t^{f_j} m^d \pmod{n}.$$

The server computes

$$S^e = t^{ef_j} m^{ed} = t^{ef_j} m \pmod{n}.$$

If this value is equal to  $m$ , the server obtains that  $f_j = 0$ . If this value is not equal to  $m$ , the server computes  $t^{eh} m$  for different values  $h$  until  $t^{eh} m = S^e \pmod{n}$ . Then  $h = f_j$ .

## 3 Protocols Using RSA Splitting

### 3.1 Capture Resilience

In a series of papers [8] and [9] MacKenzie and Reiter describe methods how to construct “capture-resilient” devices by using a network server. The idea is to protect private key operations by splitting the private key into two parts and storing one part in the device while giving the second part to the network server. Different schemes are presented for RSA encryption, RSA signatures and ElGamal signatures. For the protocols using RSA the private key  $d$  is expressed as a sum of two integers  $d_1$  and  $d_2$  such that  $d = d_1 + d_2 \pmod{\varphi(n)}$ . The device gives the half-key  $d_2$  to the server while storing  $d_1$  to its own protected memory. When needed, the device can recover  $d_1$  by asking the user to enter a password. Further, the device generates a ticket and gives the ticket and the second half-key to the server. The ticket contains necessary secret information using which the server can authenticate the device. After the initialisation the private RSA values  $p$ ,  $q$  and  $d$  are deleted.

When the user wants to generate a signature on a message, it sends a request message that contains the message and the ticket, and the authentication values encrypted using the server’s public key. Only after the server has successfully verified that the private key has not been disabled, confirmed the identity of the device and ensured that the user has inserted the correct password, the server agrees to co-operate. The server computes  $m^{d_2} \pmod n$  and returns this half-signature to the device. The device computes its half-signature  $m^{d_1} \pmod n$ . The device computes the full signature by forming the product of the half-signatures, as

$$m^{d_1}m^{d_2} = m^{d_1+d_2} = m^d \pmod n. \quad (1)$$

Since the protocol does not provide explicit authentication of the server to the client device, it is recommended that the device verifies the signature before using it. In such a manner also the server becomes authenticated.

### 3.2 Delegated Authorisation

The protocol for authorisation delegation in [13] is based on the same principle of sharing the private RSA key as the capture-resilience scheme. But the setting is different. The RSA private key is not deleted but it is kept stored in a device called the master device. The master device generates the splitting of the private exponent  $d$  and stores the first part of it to another device, called a slave device, together with some additional authentication information. The second part of the RSA key is sent to the assistant server together with necessary information, using which the server knows how to serve the slave device.

Consider the following scenario. Alice has a bank account that she can access over the Internet using her personal device. She is able to make transactions from her account by issuing digitally signed payment orders. She owns a phone and a PDA, and wants to be able to use the bank service with both. For this purpose she delegates the usage of her RSA signature key from the PDA to the phone, which does not have tamper resistant memory where the full private key could be stored. The assistant server functionality can be offered either by the phone operator or the bank. In another scenario, Bob is Alice’s son. Bob is going for a trip over the weekend and needs some

money from Alice. Alice delegates the usage of her account to Bob for the weekend and sets an upper limit to the funds she is willing to grant.

The protocol allows Alice to control the usage of her private key independently. Her PDA will inform the related assisting servers of each new delegation, or revocation of a delegation. The service does not need to know which Alice's device is used to make a transaction, or whether she has authorised somebody else to make the transaction for her. On the other hand, Alice retains the full capability of her private key with her PDA, where it is stored.

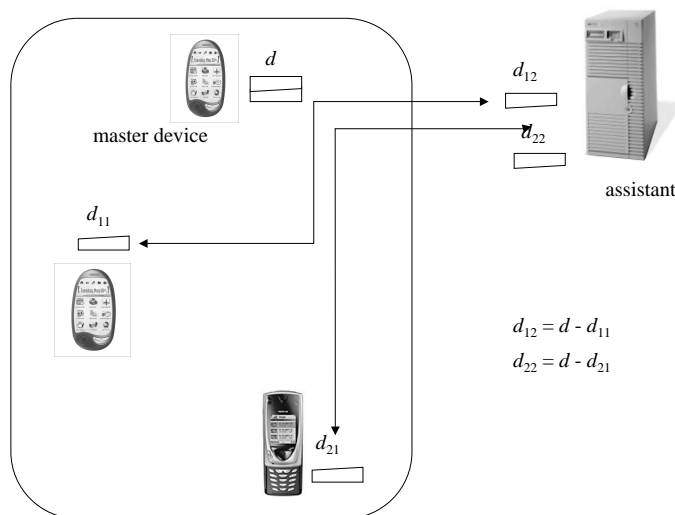


Figure 1: Initialisation of the delegated authorisation protocol

In Figure 1 it is shown how a master device delegates the private RSA capability to two slave devices. The two delegations are completely independent. The protocol also allows any of the slave devices to delegate the authorisation further to a third device. If this is not desirable, it can be prevented by the master. The slave does not have access to the full private RSA key or to the  $\varphi(n)$  value. Therefore in further delegation the slave's key share is just split into a sum of two integers. Also for the same reason the Chinese Remainder Theorem cannot be used to facilitate the private RSA computations. The private key usage is similar to the case of capture resilient devices explained in Section 3.1.

When considering server-aided computation within the delegated authorisation protocols, two different types of needs may arise depending on the scenario where it is used. First, the client device may be constrained and it would be helpful if the assisting server would do the major part of the computations. A straightforward approach would be to make the client device's half-key small. Since it is also recommended that the client-device should verify the full signature, it is then also necessary to generate the RSA key pair in such a manner that the public key is small. Small public key as such is not known to pose any threat and is frequently used in practice.

In the second scenario, the assistant serves a large number of slave clients, which perform frequent private RSA operations. Then the workload of the assistant can be reduced by making the

assistant's half-key small. In this scenario, the size of the public key need not be reduced.

These scenarios constitute special cases of the server-aided computation protocol of Section 2 with  $\ell = 2$ . The server's share is  $(w_1, w_2) = (1, d_2)$ , and the client's share is  $(f_1, f_2) = (d_1, 1)$ . Neither scenario poses any requirements to the size of the full RSA private key. Below we recall an attack on short RSA private keys, not because it would be applicable to our scenarios, but because it makes use of a mathematical technique that will be later applied to analysing the security of our scenarios.

## 4 Continued Fractions and Wiener's Attack

In this section we introduce the main mathematical tools used in this paper, which are simple continued fraction expansions and an algorithm for finding convergents. For exact mathematical treatment the reader is referred to, for example, [12]. A simple continued fraction is a chain fractions:

$$x_0 + \frac{1}{x_1 + \frac{1}{x_2 + \dots}}$$

where  $x_i, i = 0, 1, 2, \dots$  are positive integers. This expression is denoted in a more compact form by  $[x_0, x_1, x_2, \dots]$ . Each positive real number  $x$  can be represented as a continued fraction, which can be formed using the following algorithm:  $x_0$  is the integral part of  $x$ ,  $x_1$  is the integral part of the inverse of  $x - x_0$ ,  $x_2$  is the integral part of the inverse of  $\frac{1}{x - x_0} - x_1$ , and so on. Moreover, the simple continued fractions expansion of  $x$  is unique. The continued fraction expansion is finite, if there is an index  $n$  such that  $x_i = 0$ , for all  $i > n$ . Then it is denoted by  $[x_0, x_1, x_2, \dots, x_n]$ . The continued fraction expansion of a real number  $x$  is finite if and only if  $x$  is a rational number. For each positive rational number  $x$  there is an  $n$  such that  $x = [x_0, x_1, x_2, \dots, x_n]$ .

Given a simple continued fraction expansion  $[x_0, x_1, x_2, \dots]$  of a real number  $x$ , the sequence of rational numbers  $c_j, j = 1, 2, \dots$ , defined as

$$c_j = [x_0, x_1, x_2, \dots, x_{j-1}]$$

are called the convergents of  $x$ . Each  $c_j$  can be written as a rational number  $c_j = \frac{a_j}{b_j}$ , where the numerator and denominator satisfy the following recurrences:

$$a_j = \begin{cases} 1, & \text{for } j = 0 \\ x_0, & \text{for } j = 1 \\ x_{j-1}a_{j-1} + a_{j-2}, & \text{for } j \geq 2 \end{cases}$$

and

$$b_j = \begin{cases} 0, & \text{for } j = 0 \\ 1, & \text{for } j = 1 \\ x_{j-1}b_{j-1} + b_{j-2}, & \text{for } j \geq 2. \end{cases}$$

The simple continued fraction expansion of a rational number can be computed using the Euclidean algorithm. Given the continued fraction expansion, the convergents can be computed using

the above recurrences. Clearly both algorithms are polynomial in the size of the rational number of  $x$ . The size of  $x$  is the minimum of the lengths of its numerator and denominator in its reduced form.

The following theorem is very useful when one has to determine whether a rational number is a convergent of some other rational number.

**Theorem 1.** Let  $\frac{r}{s}$  and  $\frac{a}{b}$  be two positive rational numbers such that  $\gcd(r, s) = \gcd(a, b) = 1$ . If

$$\left| \frac{r}{s} - \frac{a}{b} \right| < \frac{1}{2b^2}$$

then  $\frac{a}{b}$  is one of the convergents of  $\frac{r}{s}$ .

The definitive equation of the private and public exponents of RSA is

$$ed - k\varphi(n) = 1, \tag{2}$$

which can be written also as

$$\frac{e}{\varphi(n)} - \frac{k}{d} = \frac{1}{d\varphi(n)}. \tag{3}$$

Wiener observed that  $\varphi(n)$  can be approximated using  $n$ . Then if  $d$  is sufficiently small the condition of Theorem 1 is satisfied. It follows that the fraction  $\frac{k}{d}$  can be found as a convergent of  $\frac{e}{n}$ .

The following Lemma gives an estimate of the unknown  $\varphi(n)$  approximated using  $n$ .

**Lemma 2.** Assume  $0 < p < q < n$ , where  $p$  and  $q$  are the prime factors of  $n$ , and  $q < 2p$ . Then  $n - \varphi(n) < 3\sqrt{n}$ .

**Proof.** By the definition of Euler's  $\varphi$ -function,

$$\varphi(n) = (p-1)(q-1) = pq - p - q + 1 = n - p - q + 1.$$

Because  $p < q < 2p$ , it follows that  $p < \sqrt{n}$  and  $q < 2\sqrt{n}$ , and therefore

$$n - \varphi(n) = p + q - 1 < \sqrt{n} + 2\sqrt{n} - 1 < 3\sqrt{n}.$$

■

Wiener's attack is based on the following theorem. We give the proof here for completeness.

**Theorem 3.** Let  $d < \frac{1}{3}\sqrt[4]{n}$  be the private exponent and let  $e < \varphi(n)$  be the public exponent. Then  $d$  can be found in polynomial time in  $\log n$ .

**Proof.** The lefthand side of (3) where  $\varphi(n)$  is replaced by  $n$  is estimated as follows:

$$\begin{aligned} \left| \frac{e}{n} - \frac{k}{d} \right| &= \left| \frac{ed - kn}{dn} \right| \\ &= \frac{1}{dn} |ed - k\varphi(n) + k\varphi(n) - kn| \\ &= \frac{1}{dn} |1 + k(\varphi(n) - n)| \\ &\leq \frac{1}{dn} k(n - \varphi(n)) \\ &\leq \frac{3k}{d\sqrt{n}} \end{aligned}$$

Because  $k\varphi(n) = ed - 1 < ed$  and  $e < \varphi(n)$ , it follows that  $k < d$ . Using this and the assumption we get

$$\left| \frac{e}{n} - \frac{k}{d} \right| < 3 \frac{\frac{1}{3} \sqrt[4]{n}}{d \sqrt{n}} < \frac{1}{2d^2}.$$

Now it follows from Theorem 1 that  $\frac{k}{d}$  is a convergent of  $\frac{e}{n}$ , and can be found in polynomial time.

■

The final step is to check each convergent to find the one that gives the solution for  $\varphi(n)$ . A complete algorithm to perform Wiener's attack can be found, for example, in [14].

## 5 Attacks on RSA Splitting with Small Private Exponents

The definitive equation in the RSA splitting scenarios of Section 3 is the following:

$$ed_1 + ed_2 = 1 + k\varphi(n) \quad (4)$$

Let us now consider the scenarios for server-aided computation for the capture-resilience protocol and the delegated authorisation protocol discussed at the end of Section 3. First we investigate the security in the case both  $e$  and  $d_1$  are small, and show that it is insecure. In the second case  $d_1$  is small but no requirement is set on  $e$ . Using continued fractions we show that also the second case is insecure, if  $e$  is sufficiently large. It is possible that our results can still be improved using stronger methods for finding small integer solutions, such as the LLL algorithm, in the similar manner as used by Boneh and Durfee in [2] to improve Wiener's attack.

### 5.1 Small $e$ and $d_1$

We prove the following result.

**Theorem 4.** *Assume that the private exponent  $d$  is drawn uniformly at random between 0 and  $\varphi(n)$ . Assume that  $e < \frac{1}{3} \sqrt[4]{n}$  and  $d_1$  is not essentially larger than  $e$ . Then  $d_1$  can be found with probability  $1 - \frac{1}{t}$  using  $t \frac{d_1}{e}$  trials in polynomial time in the length of  $n$ .*

**Proof.** We start by finding the value of  $k$ . We estimate the following difference of fractions:

$$\begin{aligned} \left| \frac{d_2}{n} - \frac{k}{e} \right| &= \frac{1}{en} |ed_2 - kn| \\ &= \frac{1}{en} |ed - ed_1 - k\varphi(n) + k(\varphi(n) - n)| \\ &= \frac{1}{en} |1 - ed_1 - k(n - \varphi(n))| \\ &\leq \frac{1}{en} (ed_1 + k(n - \varphi(n))). \end{aligned}$$

Since  $d < \varphi(n)$ , we have  $k < e$ . By this estimate and Lemma 2 we get

$$\left| \frac{d_2}{n} - \frac{k}{e} \right| < \frac{d_1 + 3\sqrt{n}}{n} \leq \frac{4}{\sqrt{n}} < \frac{1}{e^2},$$



using the assumption on the size of  $e$ . By Theorem 1 it follows that  $\frac{k}{e}$  is a convergent of  $\frac{d_2}{n}$ . Since the attacker knows  $e$ ,  $d_2$  and  $n$ , he can find  $k$  in polynomial time in the length of  $n$ .

It remains to find  $d_1$ . Recall the basic RSA equation (2) from where it follows that  $e^{-1} \bmod k = d \bmod k$  and therefore, the attacker can compute

$$\delta = d_1 \bmod k = (d - d_2) \bmod k = (e^{-1} \bmod k - d_2) \bmod k.$$

With probability  $1 - \frac{1}{t}$  we have  $\frac{\varphi(n)}{t} < d$ , for any  $1 < t$ . Therefore  $k\varphi(n)$ , which is about equal to  $ed$ , is larger than  $\frac{e}{t}\varphi(n)$ , with the same probability. Then  $tk > e$  and hence  $(t\frac{d_1}{e})k > d_1$ . It follows that the attacker can find the correct value of  $d_1$  in at most  $t\frac{d_1}{e}$  trials of the form  $\delta + sk$ , where  $0 \leq s < t\frac{d_1}{e}$ . ■

The last step in the attack described in the above proof can be made infeasible if  $e$  is chosen much smaller than  $d_1$ . For example if the lengths of  $e$  and  $d_1$  are 16 and 128 respectively, then the number of trials is  $t2^{112}$ .

## 5.2 Small $d_1$ and Large $e$

Now we present a result which demonstrates the insecurity of uneven splitting of the private RSA exponent in the case when the public exponent is large.

**Lemma 5.** *In the setting of Equation (4)*

$$d_2 \leq k \text{ if and only if } (\varphi(N) - e)d_2 < ed_1.$$

**Proof.** Assume first that  $d_2 \leq k$ . Then

$$ed_1 = k\varphi(n) + 1 - ed_2 \geq (\varphi(n) - e)d_2 + 1 > (\varphi(n) - e)d_2.$$

Assume then that  $ed_1 > (\varphi(n) - e)d_2$ . We get

$$(d_2 - k)\varphi(n) = d_2\varphi(n) + 1 - ed_1 - ed_2 = d_2(\varphi(n) - e) + 1 - ed_1 < 1$$

using the assumption. It follows that  $d_2 - k \leq 0$ . ■

**Theorem 6.** *Assume that in the setting of (4)  $d_1 \leq \frac{1}{3}\sqrt[4]{n}$  and that  $e$  is sufficiently large satisfying the conditions*

$$\max\left\{\frac{n}{6d_1^2}, \varphi(n) - \frac{n}{6d_1d_2}\right\} < e < \varphi(n).$$

*Further it is assumed that  $\gcd(d_1, k - d_2)$  is small. Then, given  $d_2$ ,  $e$  and  $n$ , the secret  $d_1$  can be recovered in polynomial time in the size of  $n$ .*

**Remark.** From the assumed lower bounds for  $e$  the second one is typically the maximum.

**Proof.**

$$\begin{aligned}
\left| \frac{e}{n} - \frac{k - d_2}{d_1} \right| &= \frac{|ed_1 - nk + nd_2|}{nd_1} \\
&= \frac{1}{nd_1} |1 + k\varphi(n) - ed_2 - nk + nd_2| \\
&= \frac{1}{nd_1} |1 - (k - d_2)(n - \varphi(n)) + d_2(\varphi(n) - e)|.
\end{aligned}$$

By assumption

$$d_2(\varphi(n) - e) < \frac{n}{6d_1} < ed_1.$$

Hence it follows from Lemma 5 that  $d_2 \leq k$ . Since  $e < \varphi(n)$ , it follows from 2 that  $d > k$ . Therefore  $d_1 \geq k - d_2 \geq 0$ . We use this information, the assumption that  $d_2(\varphi(n) - e) < \frac{n}{6d_1}$  and the estimate  $n - \varphi(n) \leq 3\sqrt{n}$  given by Lemma 2 to estimate further:

$$\begin{aligned}
\left| \frac{e}{n} - \frac{k - d_2}{d_1} \right| &\leq \frac{1}{nd_1} ((k - d_2)(n - \varphi(n)) + d_2(\varphi(n) - e)) \\
&\leq \frac{1}{nd_1} (3d_1\sqrt{n} + \frac{n}{6d_1}) \\
&= \frac{1}{6nd_1^2} (18d_1^2\sqrt{n} + n) \\
&= \frac{1}{6d_1^2} (2(3d_1)^2 \frac{1}{\sqrt{n}} + 1) \leq \frac{1}{2d_1^2}
\end{aligned}$$

for  $d_1 \leq \frac{1}{3}\sqrt[4]{n}$ . It follows that the reduced form of  $\frac{k-d_2}{d_1}$  is a convergent of  $\frac{e}{n}$ . Given the convergents  $\frac{a_j}{b_j}$  of  $\frac{e}{n}$ , computed in polynomial time in the size of  $n$ , the secret part  $d_1$  can be found by testing all numbers of the form  $sb_j$ , where  $1 \leq s \leq \gcd(d_1, k - d_2)$ . ■

The solution given in the proof of the theorem above becomes infeasible if  $\gcd(d_1, k - d_2)$  is large. Such parameters can be generated by selecting  $d_2$  such that  $k - d_2$  is a multiple of a large factor of  $d - k$  and at the same time  $d_1 = d - d_2$  is sufficiently small. On the other hand, if generated at random, the shares  $d_1$  and  $d_2$  will satisfy  $\gcd(d_1, k - d_2) = 1$  with probability about 0.608 (see [6], Section 4.5.2).

### 5.3 Decomposition as a Single Product

Finally let us consider the case, where the private exponent is split into two half-keys  $d_1$  and  $d_2$  in such a manner that  $d = d_1d_2 \pmod{\varphi(n)}$ . Such a splitting can be considered as a special case  $\ell = 1$ , for the server-aided computation scheme presented in Section 2. Due to the attack of Burns and Mitchell [3] it is necessary that the client verifies the result of the private key operation, before publishing the result. Therefore it is required that the public key is small.

Clearly, it would be possible to generate the RSA parameters such that  $e$  and  $d_1$  are of limited size, but only if  $\varphi(n)$  is known. Therefore further delegation without knowledge of  $\varphi(n)$  is not possible. But this scenario is always insecure, as shown by our next result.

**Theorem 7.** *Let in the context of (2)  $d = d_1 d_2 \pmod{\varphi(n)}$ . If  $e < \frac{1}{2}\sqrt[6]{n}$  and  $d_1 < \frac{1}{2}\sqrt[6]{n}$ , then  $d_1$  can be found in polynomial time in the length of  $n$ .*

**Proof.** The proof proceeds by showing that  $\frac{k}{d_1}$  is a convergent of  $\frac{ed_2}{n}$ . For this purpose we estimate their difference as follows:

$$\begin{aligned} \left| \frac{ed_2}{n} - \frac{k}{d_1} \right| &= \frac{1}{nd_1} |ed_1 d_2 - kn| \\ &= \frac{1}{nd_1} |ed - k\varphi(n) + k\varphi(n) - kn| \\ &< \frac{1}{nd_1} k(n - \varphi(n)) < \frac{3k}{d_1 \sqrt{n}}. \end{aligned}$$

Since  $d_2 < \varphi(n)$  it follows that  $k < ed_1$ . Using the assumption we estimate that  $6ed_1^2 < \sqrt{n}$ . Hence

$$\left| \frac{ed_2}{n} - \frac{k}{d_1} \right| < \frac{3e}{\sqrt{n}} < \frac{1}{2d_1^2}.$$

We see that the condition of Theorem 1 is satisfied and that  $d_1$  can be found as the denominator of one of the convergents of  $\frac{ed_2}{n}$  in polynomial time in the length of  $n$ . ■

For example, if  $n$  is a 1024-bit value, which is typical in today's applications, the splitting of the private RSA exponent is insecure already when  $e$  and  $d_1$  are 170-bit values. This is in the typical range of parameter values for server-aided computations.

## 6 Conclusion

In this paper we identified some insecure decompositions of RSA private keys in server-aided scenarios. We used continued fractions expansions. Some cases were left open. For example, we cannot tell if the case with an arbitrary public key allows secure splitting of the private key as a sum of two parts, one of which is small. An open problem to be studied is whether our results can be strengthened using methods such as the LLL algorithm used by Boneh and Durfee [2]. Such investigations would be necessary if one wants to identify the secure decompositions of the private RSA key. Although our results are negative, they reveal new properties of the RSA system.

Another approach to solving the server aid problem would be to use existing server-aided secure RSA computation protocols to facilitating the computations of either of the parties in the capture resilience protocol and the authorisation delegation protocol. This means that the server-aided protocol is applied to the private half-key, say  $d_1$ . However, certain additional conditions must then be satisfied. The server-aided computation must not use the corresponding public key, that is, the inverse of  $d_1$  modulo  $\varphi(n)$ , since the knowledge of it would allow the client to factor  $n$ . Therefore, it may not be possible to verify the result before it is delivered. Also the Chinese Remainder Theorem cannot be used. The study of suitable server-aided protocols applicable to RSA half-keys, and the exact requirements for such protocols is also left for future work.

## References

- [1] D. Boneh. Twenty Years of Attacks on the RSA Cryptosystem. *Notices of the American Mathematical Society*, 46 (1999), 203-213.
- [2] D. Boneh, G. Durfee. Cryptanalysis of RSA with Private Key  $d$  Less Than  $N^{0.292}$ . *IEEE Transactions on Information Theory*, 46 (2000), 1339-1349.
- [3] J. Burns, C. Mitchell. Parameter selection for Server-Aided RSA computation Schemes. *IEEE Transactions on Computers*, 43 (1994), 163-174.
- [4] M. J. Hinek, M. K. Low and E. Teske. On some attacks on Multi-prime RSA. In K. Nyberg and H. Heys (Eds.) *Selected Areas in Cryptography, SAC 2003*, St. Johns, Newfoundland, Canada, August 2002, LNCS 2595, Springer-Verlag 2003, 385-404.
- [5] S. M. Hong, J. B. Shin, H. Lee-Kwang and H. Yoon. A New Approach to Server-Aided Secret Computation, *Proceedings of ICISC'98*, Seoul, Korea, December 1998, 33-45.
- [6] D. E. Knuth. *The Art of Computer Programming, Vol. 2, Seminumerical Algorithms*. Addison-Wesley, 3rd edition, 1997.
- [7] A. K. Lenstra, H. W. Lenstra, Jr. and L. Lovasz. Factoring Polynomials with Rational Coefficients, *Mathematische Annalen*, 261 (1982), 515-534.
- [8] P. MacKenzie and M. K. Reiter. Networked cryptographic devices resilient to capture. In *Proceedings of the 2001 IEEE Symposium on Security and Privacy*, May 2001, 12-25.
- [9] P. MacKenzie and M. K. Reiter. Delegation of Cryptographic Servers for Capture-Resilient Devices. In *Proceedings of the 2001 ACM Conference on Computer and Communication Security*, November 2001, 10-19.
- [10] T. Matsumoto, K. Kato and H. Imai. Speeding up secret computations with insecure auxiliary devices. In S. Goldwasser (Ed.) *Advances in Cryptology - Crypto '88*, LNCS 403, Springer-Verlag 1990, 497-506
- [11] P. Pfitzmann, M. Waidner. Attacks on Protocols for Server-Aided RSA Computation. In: R. Rueppel (Ed.) *Advances in Cryptology - Eurocrypt '92*, LNCS 658, Springer-Verlag 1993, 153-162.
- [12] K. H. Rosen. *Elementary Number Theory and its Applications*. Third edition. Addison-Wesley, 1993.
- [13] S. Sovio, N. Asokan and K. Nyberg. Defining Authorization Domains Using Virtual Devices. In *IEEE 2003 Symposium on Applications and the Internet Workshops (SAINT Workshops 2003)*, January 27-31, 2003, Orlando, Florida, IEEE Computer Society Press (2003), 331-336.
- [14] D. Stinson. *Cryptography - Theory and Practice*. Second edition. Chapman&Hall/CRC, 2002.
- [15] M. Wiener. Cryptanalysis of short RSA secret exponents. *IEEE Transactions of Information Theory*, 36 (1990), 553-558.