Cryptanalysis of Server-Aided RSA Protocols with Private-Key Splitting

Thierry Mefenza, Damien Vergnaud

▶ **To cite this version:**

Thierry Mefenza, Damien Vergnaud. Cryptanalysis of Server-Aided RSA Protocols with Private-Key Splitting. The Computer Journal, 2019, 62 (8), pp.1194-1213. 10.1093/comjnl/bxz040 . hal-02082342

HAL Id: hal-02082342
https://hal.science/hal-02082342v1

Submitted on 10 May 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Cryptanalysis of Server-Aided RSA Protocols with Private-Key Splitting

Thierry Mefenza[1] and Damien Vergnaud[2]

[1] *Université Panthéon-Assas, Paris, France*
[2]*Sorbonne Université, CNRS, Laboratoire d'Informatique de Paris 6, LIP6, Paris, France*
Institut Universitaire de France
Email: thierry.mefenza-nountu@u-paris2.fr and damien.vergnaud@lip6.fr

We analyze the security and the efficiency of interactive protocols where a client wants to delegate the computation of an RSA signature given a public key, a public message and the secret signing exponent. We consider several protocols where the secret exponent is splitted using some algebraic decomposition. We first provide an exhaustive analysis of the delegation protocols in which the client outsources a single RSA exponentiation to the server. We then revisit the security of the protocols RSA-S1 and RSA-S2 that were proposed by Matsumoto, Kato and Imai in 1988. We present an improved lattice-based attack on RSA-S1 and we propose a simple variant of this protocol that provides better efficiency for the same security level. Eventually, we present the first attacks on the protocol RSA-S2 that employs the Chinese Remainder Theorem to speed up the client's computation. The efficiency of our (heuristic) attacks has been validated experimentally.

Keywords: RSA; Exponentiation outsourcing; Lattice-based cryptanalysis; Coppersmith's methods; RSA-S1; RSA-S2.

1. INTRODUCTION

Cryptographic operations are performed everywhere, from standard laptop to smart cards. The computational resources can be very limited on certain devices (like radio-frequency identification tags), and it becomes very natural, as most of the devices are online or directly connected to a powerful device (like a SIM card in a smart phone) to securely delegate some sensitive and costly operations to a device capable of carrying out cryptographic algorithms. Outsourcing cryptographic computations is a classical problem which was formalized in [1] (with protocols to outsource the computation of group exponentiation for instance).

Recently, Chevalier, Laguillaumie and Vergnaud [2] provided simple constructions for outsourcing group exponentiations in different settings (*e.g.* public/secret, fixed/variable bases and public/secret exponents) in groups of *known prime* order. They showed that their constructions are essentially optimal in terms of operations in the underlying group. Their constructions can be used in unknown order groups but it is not clear whether they are optimal in this context. In this paper, we analyze the security and the efficiency of delegation protocols for RSA exponentiations when the secret exponent is splitted using some algebraic decomposition.

1.1. Related Work

In the last 30 years, the question of how a computationally limited device may outsource group exponentiation to another, potentially malicious, but much more computationally powerful device has been a very active research topic (*e.g.* [3, 4, 5, 6, 7, 8, 9, 2]). Many solutions have been proposed and then cryptanalyzed in follow-up papers (*e.g.* [10, 11, 12, 13, 14, 2]).

In 2005, Hohenberger and Lysyanskaya [1] proposed a formal security definition for securely outsourcing computations. They showed how to securely outsource group exponentiation to two, possibly dishonest, servers that are physically separated (and do not communicate). Their protocol achieves security as long as one of them is honest (even if the computationally limited device does not know which one). In this paper, since this separation of the two servers is actually a strong assumption, we consider delegation to a *single* computationally stronger server.

In [2], Chevalier *et al.* proposed a taxonomy of exponentiation delegation protocols that covers all the practical situations : the group elements can be secret or public, variable or fixed, the exponents can be secret or public, and the result of the exponentiation can also be either public or secret. In this paper, we consider the main use case in the setting of RSA exponentiation: a client wants to delegate the computation of a signature

given a public key (N, e), a public message (or hash value of a message) m and the secret signing exponent d. By outsourcing some exponentiations to a server, the delegation protocol outputs a (public) signature $\sigma = m^d \bmod N$.

In all the proposed protocols, the basic idea is to let the server perform the main part of the computation, without giving it enough information to reconstruct the secret exponent d. Most proposed protocols are variants of two protocols (named RSA-S1 and RSA-S2) that were proposed by Matsumoto, Kato and Imai in 1988 [3]. Both schemes use a random linear decomposition of the RSA private exponent d. The goal of the present paper is to present attacks against these protocols for RSA exponentiations.

1.2. Our Contributions

Our contributions are manyfold:

- **Analysis of protocols outsourcing a single exponentiation.** As a first contribution, we analyze the delegation protocols in which the client outsources a single RSA exponentiation to the server. Such schemes generically decompose the secret exponent d among one of the following three sharing schemes: *additive splitting*, *multiplicative splitting* or *mixed splitting*. The security of the first scheme is closely related to the well-known *partial-key exposure* (e.g. see [15, 16, 17, 18]). We propose attacks for the two other paradigms that makes use of ideas proposed by Boneh and Durfee in their seminal paper on the small secret exponent cryptanalysis of RSA [19]. To counter *active attacks* against delegation protocols, Pfitzmann and Waidner [12] suggested to renew the decomposition of the secret key after each signature generation in Server-Aided RSA protocols. We also present improved attacks against these server-aided protocols when the adversary is allowed to observe several executions of the protocol (with independent random decomposition).
- **Cryptanalysis of RSA-S1.** In 2000, Merkle proposed an efficient lattice-based multi-round passive attack against RSA-S1. Merkle's attack is probabilistic and its success probability decreases rapidly if the linear decomposition of the secret exponent is limited to small dimensions. Nguyen and Shparlinski [10] proposed one year later a single-round passive attack on the RSA-S1 Server-Aided protocol which recovers the factorization of the RSA modulus when a small public exponent e is used (namely $e < N^{1/2}$). We combine these two approaches and we propose a (heuristic) lattice-based attack that works even if the public exponent is large and the linear decomposition of d is limited to small dimensions. We also propose and analyze a variant of the protocol RSA-S1 which is interesting for clients with limited memory. In this setting, the variant achieves better efficiency for the same security level.
- **Cryptanalysis of RSA-S2.** The protocol RSA-S2 employs the Chinese Remainder Theorem to speed up the client's computation. Castelluccia, Mykletun and Tsudik [7] revisited in 2006 the RSA-S2 protocol to propose a technique for re-balancing RSA-based client/server handshakes. They mentioned a *meet-in-the-middle attack* against RSA-S1 but claimed that "$\langle$ *this* $\rangle$ *attack* $\langle \dots \rangle$ *does not apply to the* $\langle$ *RSA-S2* $\rangle$ *protocol*". They proposed several security parameters for RSA-S2. We propose a simple exponential-time algorithm that break all these security parameters using multi-evaluation of polynomials.
  In [10], Nguyen and Shparlinski stated in their paper that: "*Interestingly, it seems that the* $\langle$ *lattice-based* $\rangle$ *attacks* $\langle \dots \rangle$ *do not apply to the RSA-S2 protocol.*" We also present two lattice-based attacks against RSA-S2 that break most parameters proposed by Castelluccia *et al.* (even if one updates them in order to prevent our meet-in-the-middle attack).

The efficiency of our (heuristic) attacks has been validated experimentally.

2. PRELIMINARIES

2.1. Lattices

A lattice $\mathcal{L}$ is a discrete additive subgroup of a Euclidean space. If $\mathcal{L}$ is a lattice in the n-dimensional space $\mathbb{R}^n$, then we can also define $\mathcal{L}$ as the set of all linear integer combinations of linearly independent vectors $b_1, \dots, b_k \in \mathbb{R}^n$ and:

$$\mathcal{L} = \{\sum_{j=1}^{k} a_j b_j : a_j \in \mathbb{Z}\}.$$

The set $B = \{b_1, \dots, b_k\}$ is called a basis of the lattice $\mathcal{L}$ and the integer k its dimension or rank. A lattice $\mathcal{L} \subset \mathbb{R}^n$ has full rank if its dimension equals n. If $B = \{b_1, \dots, b_k\} \subset \mathbb{R}^n$ is a basis of a lattice $\mathcal{L}$, then B can be represented by the matrix M with k rows and n columns having the basis vectors as rows and its determinant denoted $\det(\mathcal{L})$ is defined to be $\det(MM^T)^{1/2}$, where M^T is the transpose matrix of M. If $\mathcal{L}$ is full rank, then its determinant is simply $|\det(M)|$. The following lemma [20] relates the norm of the shortest vector in a lattice to its determinant.

LEMMA 2.1. *(Minkowski) Let $\mathcal{L}$ be a lattice of dimension ω. there exists a non-zero vector v with Euclidean norm:*

$$\|v\| \leqslant \sqrt{\omega}\, \det(\mathcal{L})^{1/\omega}.$$

In 1982, Lenstra, Lenstra and Lovász introduced the famous LLL algorithm [21] which on input an arbitrary

basis of a lattice $\mathcal{L}$ outputs in polynomial time a so-called reduced basis of $\mathcal{L}$ made of "short" vectors. The basis vectors of an LLL-reduced basis satisfies the following property.

LEMMA 2.2. *([21]) Let $\mathcal{L}$ be an integer lattice of dimension ω generated by k vectors with Euclidean norm at most B . In polynomial time in ω, k and $\log B$, the LLL algorithm given as input this basis of $\mathcal{L}$ outputs a* reduced *basis of $\mathcal{L}$ formed by vectors v_i, $1 \leqslant i \leqslant \omega$ that satisfy:*

$$\|v_1\| \leqslant \|v_2\| \leqslant \ldots \leqslant \|v_i\| \leqslant 2^{\frac{\omega(\omega-1)}{4(\omega+1-i)}} \det(L)^{\frac{1}{\omega+1-i}}.$$

In particular, for $i = 1$, the LLL algorithm outputs a vector v_1 with norm $\|v_1\| \leqslant 2^{\frac{\omega-1}{4}} \det(L)^{\frac{1}{\omega}}$. This approximation factor for the shortest vector of the reduced basis is exponentially large in the lattice dimension ω compared to the bound given by Lemma 2.1. However, in the following, we will often use the LLL algorithm as a "short vector oracle" (since this approximation factor translates in a constant factor for the upper-bound on the size of the roots in Coppersmith methods).

The *Gaussian heuristic* predicts the number of lattice points (for a random lattice) inside a measurable body and applied to Euclidean balls, it leads to the prediction that the length of the shortest vector in random lattice is equal to the bound given by Lemma 2.1 (up to some constant multiplicative factor). In the following, we will use the heuristic that if the norm of a specific vector is significantly smaller than the bound given by Lemma 2.1, then it is likely the shortest vector in $\mathcal{L}$ and that we can recover it (in polynomial time) using the LLL algorithm. We will apply this heuristic even if our constructed lattices are not random but it will be validated experimentally.

2.2. Coppersmith's Methods

In this section, we provide a short description of the classical Coppersmith method [22, 23] for finding small roots of a multivariate modular polynomial system of equations modulo an integer K. We refer the reader to [24] for details and proofs.

2.2.1. Problem definition.

Let $f_1(y_1, \ldots, y_n), \ldots, f_s(y_1, \ldots, y_n)$ with indeterminates y_i, for $i \in \{1, \ldots, n\}$ be irreducible multivariate polynomials defined over $\mathbb{Z}$, having a root $(x_1, \ldots, x_n)$ modulo an integer K namely $f_i(x_1, \ldots, x_n) \equiv 0 \mod K$. Our goal is to recover the desired root $(x_1, \ldots, x_n)$ in polynomial time.

To do so, we create n polynomials (see below) $\tilde{h}_i(y_1, \ldots, y_n)$, $i \in \{1, \ldots, n\}$ having as root $(x_1, \ldots, x_n)$ over the integers. We hope that the n created polynomials define an algebraic variety of dimension 0 in order to recover the desired root in polynomial time. This problem is generally intractable but becomes solvable in polynomial time (under some conditions) if the absolute value of each component of the root $(x_1, \ldots, x_n)$ is upper-bounded by some values $(X_1, \ldots, X_n)$ that depends on K and the degree of the polynomials $f_1, \ldots, f_s$.

2.2.2. Polynomials collection.

In a first step, one generates a larger collection $\mathfrak{P}$ of polynomials $\{\tilde{f}_1, \ldots, \tilde{f}_r\}$ linearly independent having $(x_1, \ldots, x_n)$ as a root modulo K^m, for some positive integer m. Usually, the technique consists in taking product of powers of the moduli K, the polynomials f_i for $i \in \{1, \ldots, s\}$ and some well-chosen monomials, such as

$$\tilde{f}_\ell = p^{m-\sum_{j=1}^{s} k_{j,\ell}} y_1^{\alpha_{1,\ell}} \cdots y_n^{\alpha_{n,\ell}} f_1^{k_{1,\ell}} \cdots f_s^{k_{s,\ell}}$$

for some positive integers $\alpha_{1,\ell}, \ldots, \alpha_{n,\ell}, k_{1,\ell}, k_{s,\ell}$. Such polynomials satisfy the relation $\tilde{f}_\ell(x_1, \ldots, x_n) \equiv 0 \mod K^m$.

2.2.3. Lattice construction.

In a second step, one denotes as $\mathfrak{M}$ the set of monomials appearing in collection of polynomials $\mathfrak{P}$, and one writes the polynomials $\tilde{f}_i(y_1 X_1, \ldots, y_n X_n)$ for $i \in \{1, \ldots, r\}$ as a vector $b_i \in (\mathbb{Z})^\omega$, where $\omega = \sharp\mathfrak{M}$. Namely If we put $\mathfrak{M} = \{m_1, \ldots, m_\omega\}$, where $m_i < m_{i+1}$ for a defined order $<$ on $\mathfrak{M}$, for $i \in \{1, \ldots, r\}$ we can write $\tilde{f}_i(y_1 X_1, \ldots, y_n X_n) = \sum_{j=1}^{\omega} c_{ij} m_j$ for some coefficients $c_{ij} \in \mathbb{Z}$ and then $b_i = (c_{i1}, \ldots, c_{i\omega}) \in (\mathbb{Z})^\omega$. Conversely, one can write a vector $v \in (\mathbb{Z})^\omega$ as a polynomial $\tilde{f}(y_1 X_1, \ldots, y_n X_n) \in \mathbb{Q}[y_1, \ldots, y_n]$. One then constructs a lattice $\mathcal{L}$ generated by the vectors $b_1, \ldots, b_r$ and computes its reduced basis using the LLL algorithm.

2.2.4. Generating new polynomials.

In a third step, one combines Lemma 2.3 below [25] and the Lemma 2.2 to get n polynomials $g_1(y_1, \ldots, y_n), \ldots, g_n(y_1, \ldots, y_n)$ having $(x_1, \ldots, x_n)$ as a root over the integers.

LEMMA 2.3. ***(Howgrave-Graham)*** *Let W be some positive integer. Let $h(y_1, \ldots, y_n)$ be a polynomial over $\mathbb{Z}$ having at most ω monomials. Suppose that:*

1. *$h(x_1, \ldots, x_n) = 0 \mod W$ for some $|x_1| < X_1, \ldots, |x_n| < X_n$ and,*
2. *$\|h(X_1 y_1, \ldots, X_n y_n)\| \leqslant \frac{W}{\sqrt{\omega}}$.*

Then $h(x_1, \ldots, x_n) = 0$ holds over the integers.

The LLL algorithm run on the lattice $\mathcal{L}$ to obtain n reduced vectors v_i, $i \in \{1, \ldots, n\}$ that we see as some polynomials $\tilde{h}_i(y_1 X_1, \ldots, y_n X_n)$, $i \in \{1, \ldots, n\}$. One can see that for $i \in \{1, \ldots, n\}$, $\tilde{h}_i(x_1, \ldots, x_n) = 0$

mod K^m, since $\tilde{h}_i$ is a linear combination of $\tilde{f}_1, \ldots, \tilde{f}_r$. Then if the following condition holds:

$$2^{\frac{r(r-1)}{4(r+1-n)}} \det(L)^{\frac{1}{r+1-n}} < \frac{K^m}{\sqrt{\omega}},$$

by Lemmas 2.2 and 2.3, $\tilde{h}_i(x_1, \ldots, x_n) = 0$, $i \in \{1, \ldots, n\}$ holds over the integers and we then obtain n polynomials having $(x_1, \ldots, x_n)$ as a root over the integers.

2.2.5. Success Condition.
In our attacks, the number of polynomials in the first step is equal to the number of monomials that appears in the collection, so $r = \omega = \sharp\mathfrak{M}$. In the analysis, we let (as usual in this setting) small terms contribute to an error term ε, and the simplified condition becomes:

$$\det(L) < K^{m\omega}.$$

We have no assurance that the n created polynomials are algebraically independent. Under the (heuristic) assumption that they define an algebraic variety of dimension 0, the previous system can be solved (e.g., using elimination techniques such as Gröbner basis) and the desired root recovered in polynomial time. In this paper, we assume that these polynomials define an algebraic variety of dimension 0 and we justify the validity of our attacks by computer experiments even when the dimension is non-zero.

2.3. Multi-exponentiation

Algorithm 1 computes the multi-exponentiation

$$\prod_{i=1}^{t} g_i^{x_i} \in \mathbb{G},$$

for $g_1, \ldots, g_t \in \mathbb{G}$ and $x_1, \ldots, x_t \in \mathbb{N}$ (see [26, 27] for details of different multi-exponentiation techniques).

Complexity: The precomputed table contains $2^t - t - 1$ non-trivial entries that require one general multiplication each. The total cost is for the precomputation phase $2^t - t - 1$ multiplications and $\ell(2^t - 1)/2^t \leqslant \ell$ multiplications on average and ℓ squarings. Although many time/memory tradeoffs are possible (e.g. using windows), for efficiency analysis, we consider in the following only delegation protocols that use this algorithm for client exponentiation.

Algorithm 1 Multi-Exponentiation Algorithm

Require: $g_1, \ldots, g_t \in \mathbb{G}$, $x_1, \ldots, x_t \in \mathbb{N}$ with $\ell = \max_{i \in \{1,\ldots,t\}} \lceil \log x_i \rceil$ and $x_j = \sum_{i=0}^{\ell-1} e_{i,j} 2^i \in \mathbb{N}$ and $e_{i,j} \in \{0,1\}$ for $i \in \{0, \ldots, \ell-1\}$ and $j \in \{1, \ldots, t\}$
Ensure: $g_1^{x_1} \cdots g_t^{x_t} \in \mathbb{G}$
  for all non-zero t-tuples $E = (E_1, \ldots, E_t) \in \{0,1\}^t$ **do**
    $g_E \leftarrow \prod_{1 \leqslant i \leqslant t} a g_i^{E_i}$ ▷ Precomputation stage
  end for
  $h \leftarrow 1_{\mathbb{G}}$
  for i from $\ell - 1$ to 0 **do**
    $h \leftarrow h^2$
    $E \leftarrow (e_{i,1}, e_{i,2}, \ldots, e_{i,t})$
    $h \leftarrow h \cdot g_E$ ▷ Multiply h by table entry
    $g_E = \prod_{1 \leqslant k \leqslant t} g_i^{e_{i,k}}$
  end for
  return h

3. DELEGATION OF A SINGLE MODULAR EXPONENTIATION

We focus in this section on protocols for outsourcing modular exponentiation (in groups of secret order) in which the client outsources only one exponentiation to the server.

More precisely, we consider the setting where the client's public key is an RSA modulus $N = pq$ where p and q are primes which are "balanced" (i.e. p and q are the same bit-size and $p, q \simeq N^{1/2}$) together with a public exponent e such that $e \simeq N^\alpha$ for some $\alpha \geqslant 0$. In the rest of the paper, p and q have approximately the same sizes and $p, q \simeq N^{1/2}$ simply means that p and q are the same bit-size. Most standard implementations of RSA use very small public exponent such as $e \in \{3, 5, 17, 257, 65537\}$ and the case $\alpha \simeq 0$ is of very strong practical importance. The client's secret key is defined by $d = e^{-1} \bmod \varphi(N)$ where $\varphi(N) = (p-1)(q-1)$ is the Euler totient function.

The client has to compute $c = m^d \bmod N$ for some variable $m \in \mathbb{Z}_N^*$ (which is assumed to be public) and it is allowed to query the server on a single triple (m, r, N) to get $c' = m^r \bmod N$. Using its knowledge of d and c', the client eventually outputs c. Using only the multiplicative structure of $\mathbb{Z}_N$, the client can compute c in essentially three different ways:

- $c = m^{d_1} \cdot c' \bmod N$, which corresponds to an *additive splitting* of the secret key as $d = d_1 + r \bmod \varphi(N)$;
- $c = c'^{d_1} \bmod N$, which corresponds to a *multiplicative splitting* of the secret key as $d = r \cdot d_1 \bmod \varphi(N)$;
- $c = m^{d_1} c'^{d_2} \bmod N$, which corresponds to a *mixed splitting* of the secret key as $d = d_1 + r \cdot d_2 \bmod \varphi(N)$.

In order to be computationally efficient for the client, these protocols should use small d_1 and d_2. However, it is worth noting that the value r can be picked arbitrarily (and in particular it may be larger than $\varphi(N)$). We present attacks on these three protocols when d_1 and d_2 are too small.

3.1. Server-Aided Protocol with Additive Splitting

In the *server-aided protocol with additive splitting* (see Fig. 1), the secret d is decomposed as $d = d_1 + r \bmod \varphi(N)$, where $d_1 \simeq N^\delta$ is kept secret by the client and

$$r = [(d - d_1) \mod \varphi(N)] + k\varphi(N)$$

is known where k is some unknown integer with $k \simeq N^{\beta_1}$.

Putting $d^\star = d_1 + r$, then $d = d^\star \bmod \varphi(N)$ and $d^\star \simeq N^\beta$, with $\beta = 1 + \beta_1$. The adversary knows $r \simeq N^\beta$ that is an approximation of $d^\star \simeq N^\beta$ and d_1 has to be found. This problem is actually very close to the so-called *partial key exposure attacks* which have been widely studied (e.g. see for instance [15, 16, 17, 18, 28, 29]). We can simply apply the following result due to Joye and Lepoint [15] which is the best known attack for the partial key exposure with *exponent blinding*:

THEOREM 3.1 (Joye-Lepoint). *Let $N = pq$ be a product of two primes, $e \simeq N^\alpha$ be an integer, $d = e^{-1} \bmod \varphi(N)$ and $d^\star = d + \ell\varphi(N) \simeq N^\beta$ be an integer. If $d^\star = \tilde{d} + d_0$ with $\tilde{d}$ known, $d_0 \simeq N^\delta$ unknown, then for sufficiently large N, there exists a (heuristic) polynomial-time algorithm that can computes all of $d^\star$ provided that:*

$$\delta < \begin{cases} \alpha + \beta - 1 & \text{for } 1 < \alpha + \beta < 3/2 \\ \frac{\alpha+\beta-\sqrt{4(\alpha+\beta)^2 - 6(\alpha+\beta)}}{3} & \text{for } 3/2 < \alpha + \beta < 2 \end{cases}$$

Applying Theorem 3.1 with $d^\star = d_1 + r \simeq N^\beta$, $\beta = \beta_1 + 1$, and $d_1 \simeq N^\delta$, we obtain readily the following result (see also Figure 2):

COROLLARY 3.1. *There exists a (heuristic) polynomial-time passive adversary against the server-aided protocol with additive splitting that can recover the client secret d (for sufficiently large RSA modulus) provided that:*

$$\delta < \begin{cases} \beta_1 + \alpha & \text{for } 0 < \alpha + \beta_1 < 1/2 \\ \frac{\alpha+\beta_1+1-\sqrt{4(\alpha+\beta_1)^2 + 2(\alpha+\beta_1) - 2}}{3} & \text{for } 1/2 < \alpha + \beta_1 < 1 \end{cases}$$

3.2. Server-Aided Protocol with Multiplicative Splitting

In the *server-aided protocol with multiplicative splitting* (see Fig. 3), the secret d is decomposed as

$$d = d_1 \cdot r \bmod \varphi(N),$$

where $d_1 \simeq N^\delta$ is kept secret by the client and

$$r = (d/d_1) \mod \varphi(N) + k\varphi(N)$$

is known where k is some unknown integer with $k \simeq N^{\beta_1}$.

The goal of the adversary is to recover d_1. There exists $k^\star \simeq N^{\alpha+\beta_1+\delta}$ such that:

$$e \cdot r \cdot d_1 = 1 + k^\star \cdot (N + (1 - p - q)).$$

In the following, we present two lattice-based attacks using this equation for the two settings when the public exponent e is "small" (i.e. $\alpha < 1$) or arbitrary.

3.2.1. Case where e is "small" ($\alpha < 1$)
We put $e' = er \simeq N^{1+\alpha+\beta_1}$ and we have:

$$1 + k^\star(N + (1 - p - q)) = 0 \bmod e'.$$

The polynomial $f(x, y) = 1 + x(N + y)$ has as root $X_0 = (k^\star, 1 - p - q)$ modulo e', with $|k^\star| \leqslant X = N^{\alpha+\beta_1+\delta}$ and $|1 - p - q| \leqslant Y$ with $Y \simeq N^{1/2}$ (for large N). We use the linearization $u = xy + 1$ as in the attack on RSA with small secret exponent proposed by Boneh and Durfee [19] and simplified by Herrmann and May [30]. We thus obtain the linear polynomial $\bar{f}(u, x) = u + Nx$. We use the same collection of polynomials as in [30]:

$$\begin{cases} x^i \bar{f}^j (e')^{m-j} & \text{for } i, j \geqslant 0,\ i + j \leqslant m \\ y^j \bar{f}^i (e')^{m-i} & \text{for } 1 \leqslant j \leqslant t,\ \lfloor \frac{m}{t} \rfloor j \leqslant i \leqslant m \end{cases}$$

With $t = \tau m$, $\tau \leqslant 1$ (for some parameter τ to be optimized) having as root $X_0 = (k^\star, 1 - p - q, k^\star(1 - p - q) + 1)$ modulo $(e')^m$, with $|k^\star(1 - p - q) + 1| \leqslant U = N^{\alpha+\beta_1+\delta+1/2}$. Thus we define a lattice $\mathcal{L}$ generated the coefficient vector of each polynomial $\tilde{f}(Xx, Yy, Uu)$, for $\tilde{f}$ in the collection of polynomials $\mathfrak{P}$. One can put an appropriate order on the set $\mathfrak{P}$ (see [30]) for details and obtain a lower triangular matrix with the diagonal elements

$$X^i U^j (e')^{m-j} \quad (\text{for } i, j \geqslant 0 \text{ and } i + j \leqslant m) \text{ and}$$

$$U^i Y^j (e')^{m-i} \quad \left(\text{for } 1 \leqslant j \leqslant t \text{ and } \left\lfloor \frac{m}{t} \right\rfloor j \leqslant i \leqslant m\right).$$

The lattice $\mathcal{L}$ is of dimension

$$\omega = \sum_{i=0}^{m} \sum_{j=0}^{m-i} 1 + \sum_{j=1}^{\tau m} \sum_{i=\frac{m}{t}j}^{m} 1 = \left(\frac{1}{2} + \frac{\tau}{2}\right) m^2 + o(m^2)$$

and has determinant

$$\det(\mathcal{L}) = X^{s_x} Y^{s_y} U^{s_u} (e')^{s_{e'}},$$

where

$$s_x = \sum_{i=0}^{m} \sum_{j=0}^{m-i} i = \frac{1}{6} m^3 + o(m^3),$$

$$s_y = \sum_{j=1}^{\tau m} \sum_{i=\lfloor \frac{m}{t} \rfloor j}^{m} j = \frac{\tau^2}{6} m^3 + o(m^3),$$

$$s_u = \sum_{i=0}^{m} \sum_{j=0}^{m-i} j + \sum_{j=1}^{\tau m} \sum_{i=\lfloor \frac{m}{t} \rfloor j}^{m} i = \left(\frac{1}{6} + \frac{\tau}{3}\right) m^3 + o(m^3),$$

Client		Server
$\mathbf{pk} = (N, e)$, $\mathbf{sk} = (d, \varphi(N))$		
$d_1 \xleftarrow{R} \{0, \ldots, N^{\delta}\}$; $k \xleftarrow{R} \{0, \ldots, N^{\beta_1}\}$		
$r \leftarrow d - d_1 + k \cdot \varphi(N)$	$\xrightarrow{m, r, N}$	
	$\xleftarrow{c'}$	$c' \leftarrow m^r \bmod N$
$c \leftarrow c \cdot m^{d_1} \bmod N$		
Output c		

FIGURE 1. Server-Aided RSA Protocol with Additive Splitting

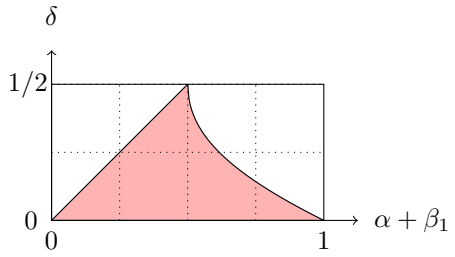

FIGURE 2. Cryptanalysis of Server-Aided RSA Protocol with Additive Splitting

$$s_{e'} = \sum_{i=0}^{m} \sum_{j=0}^{m-i} (m-j) + \sum_{j=1}^{\tau m} \sum_{i=\lfloor \frac{m}{t} \rfloor j}^{m} (m-i)$$
$$= \left(\frac{1}{3} + \frac{\tau}{6}\right) m^3 + o(m^3).$$

For large m, we can find all the solutions of the modular equation $f(x, y) = 0 \bmod e'$ if the following condition holds:

$$\det(\mathcal{L}) < ((e')^m)^{\omega},$$

that is

$$\left(\frac{1}{3} + \frac{\tau}{3}\right)(\alpha + \beta_1 + \delta) + \frac{1}{2}\left(\frac{1}{6} + \frac{\tau}{3} + \frac{\tau^2}{6}\right) < \left(\frac{1}{6} + \frac{\tau}{3}\right)(1 + \beta_1 + \alpha).$$

To maximize δ, we choose $\tau_{max} = 1 - 2\delta$ and obtain the inequality

$$2\delta^2 - 4\delta + 1 - \alpha - \beta_1 > 0$$

and the following result:

THEOREM 3.2. *There exists a (heuristic) polynomial-time passive adversary against the server-aided protocol with multiplicative splitting that can recover the client secret d (for sufficiently large RSA modulus) provided that:*

$$\delta < 1 - \sqrt{\frac{1 + \alpha + \beta_1}{2}}$$

REMARK 1. In our result, we used as usual in the setting of Coppersmith's methods the simplified success condition rather than the original condition for sufficiently large modulus N. For N of bit-length larger than 256, the bound of Theorem 3.2 can be reached for $m \geqslant 6$. For N of bit-length respectively 128, the bound can be reached for $m \geqslant 8$. Thus by sufficiently large modulus, we mean that the bit-length of the modulus is at least 128, which is always the case in practical applications of cryptography.

REMARK 2. We do not have a passive attack against the protocol with multiplicative splitting if $\alpha + \beta_1 > 1$ (in particular if e is of the same size as N).

3.2.2. Case where e is arbitrary.
In order to attack the protocol for arbitrary public exponent e, we have to consider a stronger attack scenario. As mentioned in the introduction, it has been suggested several times to renew the decomposition of the secret exponent for each invocation of the protocol. In this paragraph, we present an attack against the server-aided protocol with multiplicative splitting when the adversary is allowed to observe $k \geqslant 2$ different executions of the protocol (with independent random values).

We thus suppose that d is decomposed k times as $d = d_i \cdot r_i$ with $d_i \simeq U = N^{\delta}$ for $i \in \{1, \ldots, k\}$, with $r_i = (d/d_i \mod \varphi(N)) + k_i\varphi(N) \simeq N^{1+\beta_1}$ where $k_i \simeq N^{\beta_1}$. For $j \in \{2, \ldots, k\}$, we have:

$$d_j r_j - d_1 r_1 = 0 \bmod \varphi(N),$$

thus for $j \in \{2, \ldots, k\}$, there exists $k_j^{\star} \simeq N^{\beta_1 + \delta}$ such that:

$$d_j r_j - d_1 r_1 + k_j^{\star}(p + q - 1) = 0 \bmod N.$$

Then for $j \in \{2, \ldots, k\}$, we obtain a polynomial $f_j(y_1, \ldots, y_k, z_2, \ldots, z_k)$ having as root $X = (d_1, \ldots, d_k, k_2^{\star}(p+q-1), \ldots, k_k^{\star}(p+q-1))$ modulo N with:

$$f_j = r_j y_j - y_1 r_1 + z_j \bmod N$$

We put $f_j' = f_j - z_j$ for $j \in \{2, \ldots, k\}$, $K = \lfloor N^{1/2+\beta_1} \rfloor$ and $\omega = 2k - 1$ and we define the ω-dimensional lattice $\mathcal{L}$ spanned by the rows of the following matrix:

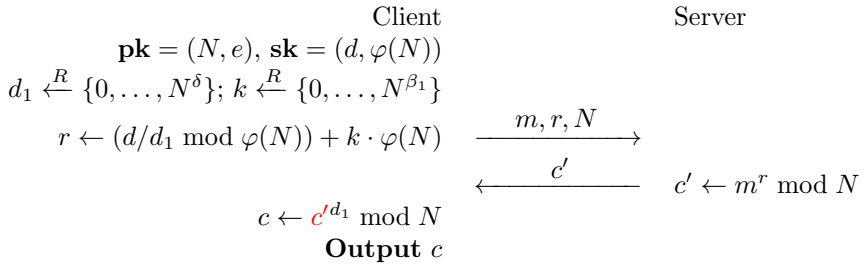

FIGURE 3. Server-Aided RSA Protocol with Multiplicative Splitting

$$\mathcal{A} = \left(\begin{array}{ccc|ccc|c} K & & & f_2' & \cdots & f_k' & y_1 \\ & K & & \downarrow & \downarrow & \downarrow & y_2 \\ & & \ddots & & & & \vdots \\ & & K & & & & y_k \\ \hline 0 & & & N & & & \\ & & & & \ddots & & \\ & & & & & N & \end{array}\right)$$

The right-hand side is formed by all vectors coming from the set of polynomials $\{f_2', \ldots, f_k'\}$. We have $\det(\mathcal{L}) = K^k N^{k-1}$. The lattice $\mathcal{L}$ contains the vector

$$v = (Kd_1, \ldots, Kd_k, k_2^\star(p+q-1), \ldots, k_k^\star(p+q-1))$$

with the Euclidean norm $\|v\| = O(\sqrt{\omega} UK)$. If we can find the vector v, then we will be able to recover the unknowns d_j for $j \in \{1, \ldots, k\}$. By Lemma 2.1 and our heuristic from Section 2.1, the vector v is likely to be the shortest vector in the lattice $\mathcal{L}$ if the following condition holds:

$$\|v\| \leqslant \sqrt{\omega}\det(\mathcal{L})^\omega,$$

which is equivalent to

$$U \leqslant N^{\frac{(k-1)(1/2-\beta_1)}{2k-1}}$$

Therefore, by using the LLL algorithm (the LLL algorithm is also used in the forthcoming results) we can heuristically recover $(d_1, \ldots, d_k)$ with $d_i < N^\delta$ as long as:

$$\delta < \frac{(k-1)(1/2-\beta_1)}{2k-1}.$$

THEOREM 3.3. *There exists a (heuristic) polynomial-time passive adversary against the server-aided protocol with multiplicative splitting that can recover the client secret d (for sufficiently large RSA modulus) after eavesdropping $k \geqslant 2$ different executions of the protocol provided that:*

$$\delta < \frac{(k-1)(1/2-\beta_1)}{2k-1}.$$

The results from Theorems 3.2 and 3.3 are illustrated on Figure 4 (where the red zones indicate the values δ for which the protocol with multiplicative splitting is insecure).

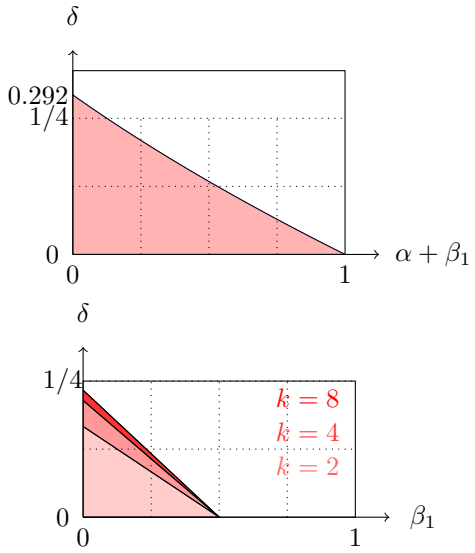

FIGURE 4. Cryptanalysis of Server-Aided RSA Protocol with Multiplicative Splitting (*left:* small $e \simeq N^\alpha$; *right:* arbitrary e, $k \in \{2, 4, 8\}$ rounds)

3.3. Server-Aided Protocol with Mixed Splitting

In the *server-aided protocol with mixed splitting* (see Fig. 5), the secret d is decomposed as

$$d = d_1 + d_2 \cdot r \bmod \varphi(N),$$

where $d_1 \simeq N^{\delta_1}$, $d_2 \simeq N^{\delta_2}$ are kept secret unknown and

$$r = (d - d_1)/d_2 \mod \varphi(N) + k \cdot \varphi(N)$$

is known where k is some unknown integer with $k \simeq N^{\beta_1}$. The goal of the adversary is to recover d_1 and d_2. There exists $k^\star \simeq N^{\alpha+\beta_1+\delta_2}$ such that:

$$e(d_1 + d_2 r) = 1 + k^\star(N + (1 - p - q)).$$

In the following, we present two lattice-based attacks using this equation for the two setting when the public exponent e is "small" (i.e. $\alpha < 1$) or arbitrary.

3.3.1. Case where e is "small" ($\alpha < 1$)
We put $e' = e \cdot r \simeq N^{1+\alpha+\beta_1}$ and we have:

$$1 + k^\star(N + (1 - p - q)) - ed_1 = 0 \bmod e'.$$

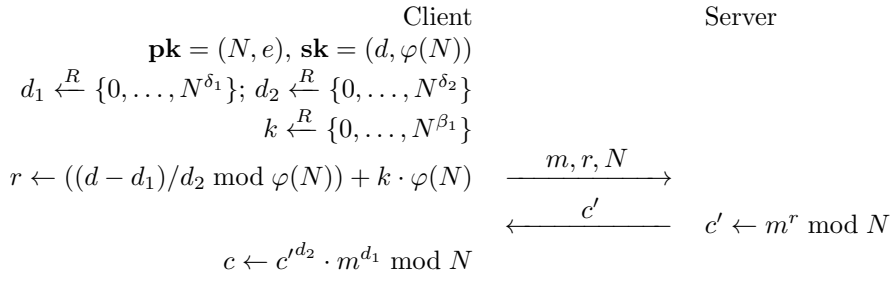

FIGURE 5. Server-Aided RSA Protocol with Mixed Splitting

The trivariate polynomial $f(x,y,z) = 1+x(N+y)-ez$ has as root $X_0 = (k^\star, 1-p-q, d_1)$ modulo e', with $|k^\star| \leqslant X = N^{\alpha+\beta_1+\delta_2}$, $|1-p-q| \leqslant Y = N^{1/2}$ and $|d_1| \leqslant Z = N^{\delta_1}$. We consider the following collection of polynomials:

$$\begin{cases} x^i z^k f^j (e')^{m-j} & \text{for } i,j,k \geqslant 0,\ i+j+k \leqslant m \\ y^j z^k f^i (e')^{m-i} & \text{for } i,k \geqslant 0,\ i+k \leqslant m;\ 1 \leqslant j \leqslant t \end{cases}$$

With $t = \tau m$ (for some parameter τ to be optimized). If we proceed as in the previous section (by putting an appropriate order an the set of monomials and the set of polynomials), we obtain a lattice $\mathcal{L}$ of dimension

$$\omega = \sum_{i+j+k\leqslant m} 1 + \sum_{i+k\leqslant m; 1\leqslant j\leqslant t} 1 = \left(\frac{1}{6}+\frac{\tau}{2}\right) m^3 + o(m^3)$$

and $\det(\mathcal{L}) = X^{s_x} Y^{s_y} Z^{s_z} (e')^{s_{e'}}$, where:

$$\begin{aligned} s_x &= \sum_{i+j+k\leqslant m} i+j + \sum_{i+k\leqslant m; 1\leqslant j\leqslant t} i \\ &= \left(\frac{1}{12}+\frac{\tau}{6}\right) m^4 + o(m^4), \\ s_y &= \sum_{i+j+k\leqslant m} j + \sum_{i+k\leqslant m; 1\leqslant j\leqslant t} i+j \\ &= \left(\frac{1}{24}+\frac{\tau^2}{4}+\frac{\tau}{6}\right) m^4 + o(m^4), \\ s_z &= \sum_{i+j+k\leqslant m} k + \sum_{i+k\leqslant m; 1\leqslant j\leqslant t} k \\ &= \left(\frac{1}{24}+\frac{\tau}{6}\right) m^4 + o(m^4), \end{aligned}$$

and

$$\begin{aligned} s_{e'} &= \sum_{i+j+k\leqslant m} m-i + \sum_{i+k\leqslant m; 1\leqslant j\leqslant t} m-i \\ &= \left(\frac{1}{8}+\frac{\tau}{3}\right) m^4 + o(m^4). \end{aligned}$$

We can thus find all the solutions of the modular equation $f(x,y,z) = 0 \bmod e'$ if $\det(\mathcal{L}) < ((e')^m)^\omega$, i.e.

$$\begin{aligned} &\left(\frac{1}{12}+\frac{\tau}{6}\right)(\alpha+\beta_1+\delta_2) + \frac{1}{2}\left(\frac{1}{24}+\frac{\tau}{6}+\frac{\tau^2}{4}\right) \\ &\quad +\delta_1\left(\frac{1}{24}+\frac{\tau}{6}\right) < \left(\frac{1}{24}+\frac{\tau}{6}\right)(1+\beta_1+\alpha). \end{aligned}$$

To maximize δ_1 and δ_2, we choose $\tau_{max} = \frac{1-2(\delta_1+\delta_2)}{3}$ and obtain the inequality

$$-24(\delta_1^2+\delta_2^2) - 48\delta_1\delta_2 + 42\delta_1 + 60\delta_2 - 15 + 18(\alpha+\beta_1) < 0.$$

THEOREM 3.4. *There exists a (heuristic) polynomial-time passive adversary against the server-aided protocol with mixed splitting that can recover the client secret d (for sufficiently large RSA modulus) provided that:*

$$-24(\delta_1^2+\delta_2^2) - 48\delta_1\delta_2 + 42\delta_1 + 60\delta_2 - 15 + 18(\alpha+\beta_1) < 0.$$

Furthermore if $\delta_1 = \delta_2 = \delta$, we obtain the following corollary:

COROLLARY 3.2. *There exists a (heuristic) polynomial-time passive adversary against the server-aided protocol with mixed splitting (with $\delta_1 = \delta_2 = \delta$) that can recover the client secret d (for sufficiently large RSA modulus) provided that:*

$$\delta < \frac{17 - \sqrt{129 + 192(\alpha+\beta_1)}}{32}.$$

Theorem 3.4 can be slightly improved by following the technique of [31] for attacking the dual RSA. If $\delta_1 = \delta_2 = \delta$, we show that there exists a (heuristic) polynomial-time passive adversary that can recover the client secret d (for sufficiently large RSA modulus) provided that:

$$\delta < (6 - \sqrt{15 + 24(\alpha+\beta_1)})/12.$$

We put the linearization $u = xy + 1$ as in the cryptanalysis of the dual RSA [31] and we obtain the linear polynomial $\bar{f}(u,x,z) = u + Nx - ez$. We use the same collection of polynomials as in [31]:

$$\begin{cases} x^i z^k \bar{f}^j (e')^{m-j} & \text{for } i,j,k \geqslant 0,\ i+j+k \leqslant m \\ y^j z^{k-i} \bar{f}^i (e')^{m-i} & \text{for } 1 \leqslant j \leqslant t,\ \lfloor \frac{m}{t} \rfloor j \leqslant k \leqslant m, \\ & 0 \leqslant i \leqslant k. \end{cases}$$

With $t = \tau m$, $\tau \leqslant 1$ (for some parameter τ to be optimized) having as root $X_0 = (k^\star(1-p-q)+1, k^\star, 1-p-q, d_1)$ modulo $(e')^m$, with $|k^\star| \leqslant X = N^{\alpha+\beta_1+\delta_2}$, $|1-p-q| \leqslant Y = N^{1/2}$, $|d_1| \leqslant Z = N^{\delta_1}$ and $|k^\star(1-p-q)+1| \leqslant U = XY$. If we put an appropriate order on the set of monomials and on the set of polynomials, we obtain a lattice $\mathcal{L}$ whose matrix is a lower triangular matrix with diagonal elements:

$$X^i Z^k U^j (e')^{m-j} \quad \text{for } i,j,k \geqslant 0,\ i+j+k \leqslant m,$$

$$Y^j Z^{k-i} U^i (e')^{m-i} \qquad \text{for } 1 \leqslant j \leqslant t,\ \left\lfloor \frac{m}{t} \right\rfloor j \leqslant k \leqslant m, \quad 0 \leqslant i \leqslant k.$$

The lattice $\mathcal{L}$ is of dimension

$$\omega = \sum_{i+j+k \leqslant m} 1 + \sum_{j=1}^{t} \sum_{k=\lfloor \frac{1}{\tau} \rfloor j}^{m} \sum_{i=0}^{k} 1 = \left(\frac{1}{6} + \frac{\tau}{3}\right) m^3 + o(m^3)$$

and has determinant $\det(\mathcal{L}) = U^{s_u} X^{s_x} Y^{s_y} Z^{s_z} (e')^{s_{e'}}$, where:

$$s_u = \sum_{i+j+k \leqslant m} j + \sum_{j=1}^{t} \sum_{k=\lfloor \frac{1}{\tau} \rfloor j}^{m} \sum_{i=0}^{k} i = \left(\frac{1}{24} + \frac{\tau}{8}\right) m^4 + o(m^4),$$

$$s_x = \sum_{i+j+k \leqslant m} i = \frac{1}{24} m^4 + o(m^4),$$

$$s_y = \sum_{j=1}^{t} \sum_{k=\lfloor \frac{1}{\tau} \rfloor j}^{m} \sum_{i=0}^{k} j = \frac{\tau^2}{8} m^4 + o(m^4),$$

$$s_z = \sum_{i+j+k \leqslant m} k + \sum_{j=1}^{t} \sum_{k=\lfloor \frac{1}{\tau} \rfloor j}^{m} \sum_{i=0}^{k} k - i = \left(\frac{1}{24} + \frac{\tau}{8}\right) m^4 + o(m^4),$$

$$s_{e'} = \sum_{i+j+k \leqslant m} m - i + \sum_{j=1}^{t} \sum_{k=\frac{1}{\tau} j}^{m} \sum_{i=0}^{k} m - i = \left(\frac{1}{8} + \frac{5\tau}{24}\right) m^4 + o(m^4).$$

We can thus find all the solutions of the modular equation $\bar{f}(u,x,y,z) = 0 \bmod e'$ if the following condition holds:

$$\det(\mathcal{L}) < ((e')^m)^\omega,$$

that is

$$\left(\frac{1}{12} + \frac{\tau}{8}\right)(\alpha + \beta_1 + \delta_2) + \frac{1}{2}\left(\frac{1}{24} + \frac{\tau}{8} + \frac{\tau^2}{8}\right) + \delta_1 \left(\frac{1}{24} + \frac{\tau}{8}\right) < \left(\frac{1}{24} + \frac{\tau}{8}\right)(1 + \beta_1 + \alpha).$$

To maximize δ_1 and δ_2, we choose $\tau_{max} = \frac{1-2(\delta_1+\delta_2)}{2}$ and obtain the inequality

$$-24(\delta_1^2+\delta_2^2) - 48\delta_1\delta_2 + 40\delta_1 + 56\delta_2 - 14 + 16(\alpha+\beta_1) < 0.$$

THEOREM 3.5. *There exists a (heuristic) polynomial-time passive adversary against the server-aided protocol with mixed splitting that can recover the client secret d (for sufficiently large RSA modulus) provided that:*

$$-24(\delta_1^2+\delta_2^2) - 48\delta_1\delta_2 + 40\delta_1 + 56\delta_2 - 14 + 16(\alpha+\beta_1) < 0.$$

Furthermore if $\delta_1 = \delta_2 = \delta$, we obtain the following corollary:

COROLLARY 3.3. *There exists a (heuristic) polynomial-time passive adversary against the server-aided protocol with mixed splitting (with $\delta_1 = \delta_2 = \delta$) that can recover the client secret d (for sufficiently large RSA modulus) provided that:*

$$\delta < \frac{6 - \sqrt{15 + 24(\alpha + \beta_1)}}{12}.$$

Theorem 3.5 slightly improved Theorem 3.4 but the improvement is mainly theoretical and in the following we have only implemented the attack corresponding to Theorem 3.4.

3.3.2. Experimental Results

We have implemented the attack in Sage 7.6 on a MacBook Air laptop computer (2,2 GHz Intel Core i7, 4 Gb RAM 1600 MHz DDR3, Mac OSX 10.10.5) for a 2048-bit modulus N and for $e = 3$ ($\alpha \simeq 0$). We ran experiments for several values of β_1, namely $\beta_1 \in \{0.002, 0.25, 0.5\}$. We suppose that $d_1, d_2 < N^\delta$ and the table below lists the theoretical bound

$$\delta_{\text{theo}} = \frac{17 - \sqrt{129 + 192\beta_1}}{32}$$

and the experimental bound δ_{exp}. The size of the manipulated integers grows with β_1. After the Gröbner basis computations, we obtained a system of polynomials of dimension 1 but were always able to find the desired root (by factoring the first polynomial of the Gröbner with gives $p + q - 1$ and thus N's factorization). We run 2^7 experiments for each choice of parameters and Table 1 gives the average running times (in seconds) of the LLL algorithm and the Gröbner basis computation.

β_1	δ_{theo}	δ_{exp}	m	dimension	LLL time(s)	Gröbner basis time(s)
0.002	0.175	0.17	5	77	8.89	0.026
0.25	0.115	0.115	4	50	1.51	0.011
0.5	0.062	0.06	4	65	24.07	14.97

TABLE 1. Experimental Results (Theorem 3.4) – Average running times (in seconds) of the LLL algorithm and the Gröbner basis computation

3.3.3. Case where e is arbitrary.
In order to attack the protocol for arbitrary public exponent e, we consider again the stronger attack scenario where the adversary is allowed to observe $k \geqslant 2$ different executions of the protocol (with independent random values). We suppose that d is decomposed k times as $d = d_1^{(i)} + r_i d_2^{(i)} \bmod \varphi(N)$ with $d_1^{(i)} \simeq N^{\delta_1}$, $d_2^{(i)} \simeq N^{\delta_2}$ unknown for $i \in \{1, \ldots, k\}$, and $r_i = (d - d_1^{(i)})/d_2^{(i)} \bmod \varphi(N) + k_i\varphi(N) \simeq N^{1+\beta_1}$ where $k_i \simeq N^{\beta_1}$ is unknown. For $i \in \{2, \ldots, k\}$, we have:

$$d_1^{(i)} + r_i d_2^{(i)} - d_1^{(1)} + r_1 d_2^{(1)} = 0 \bmod \varphi(N),$$

thus for $i \in \{2, \ldots, k\}$, there exists $k_i^\star \simeq N^{\beta_1+\delta_2}$ such that:

$$d_1^{(i)} + r_i d_2^{(i)} - d_1^{(1)} + r_1 d_2^{(1)} + k_i^\star(p + q - 1) = 0 \bmod N.$$

Then using the same technique as for e arbitrary in the multiplicative splitting, we can recover $(d_1^{(1)}, d_2^{(1)} \ldots, d_1^{(k)}, d_2^{(k)})$ as long as:

$$k(\delta_1 + \delta_2) + (1/2 + \beta_1 + \delta_2)(k - 1) < k - 1,$$

that is:

$$k\delta_1 + (2k - 1)\delta_2 < (k - 1)(1/2 - \beta_1).$$

THEOREM 3.6. *There exists a (heuristic) polynomial-time passive adversary against the server-aided protocol with mixed splitting that can recover the client secret d (for sufficiently large RSA modulus) after eavesdropping $k \geqslant 2$ different executions of the protocol provided that:*

$$k\delta_1 + (2k - 1)\delta_2 < (k - 1)(1/2 - \beta_1).$$

COROLLARY 3.4. *There exists a (heuristic) polynomial-time passive adversary against the server-aided protocol with mixed splitting (with $\delta_1 = \delta_2 = \delta$) that can recover the client secret d (for sufficiently large RSA modulus) after eavesdropping $k \geqslant 2$ different executions of the protocol provided that:*

$$\delta < \frac{k - 1}{3k - 1}(1/2 - \beta_1).$$

Our results are illustrated on Figure 6 (where the red zones indicate the values δ for which the protocol with multiplicative splitting is insecure).

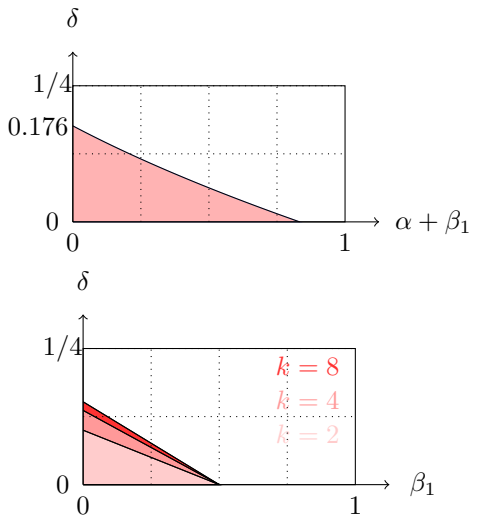

FIGURE 6. Cryptanalysis of Server-Aided RSA Protocol with Mixed Splitting (*left:* small $e \simeq N^\alpha$; *right:* arbitrary e, $k \in \{2, 4, 8\}$ rounds)

4. CRYPTANALYSIS OF RSA-S1 PROTOCOL

4.1. Description of a RSA-S1

In 1988, Matsumoto, Kato and Imai [3] proposed the following protocol, known as RSA-S1, in which a client computes an RSA exponentiation with secret exponent (e.g. a signature), with the help of an untrusted powerful server. In the following, we assume that the server is honest-but-curious (i.e. we consider only passive adversaries).

- **Step 1.** The client picks uniformly at random $n \geqslant 1$ "small" integers $d_i \in \mathbb{Z}$, for $i \in \{1, \ldots, n\}$ with $d_i \in \{0, \ldots, U\}$ for some integer U.
- **Step 2.** The client picks uniformly at random n elements $r_i \in \mathbb{Z}_N$, for $i \in \{1, \ldots, n\}$ from the set of vectors satisfying the congruence:

$$d_1 \cdot r_1 + \cdots + d_n \cdot r_n = d \bmod \varphi(N).$$

- **Step 3.** The client sends the $(n + 2)$-tuple $(m, r_1, \ldots, r_n, N)$ to the server.
- **Step 4.** The server computes and sends to the client $z_i = m^{r_i} \bmod N$, for $i \in \{1, \ldots, n\}$.

- **Step 5.**. The client computes

$$\prod_{i=1}^{n} z_i^{d_i} = z_1^{d_1} \dots z_n^{d_n} \bmod N$$

which is equal to $m^d \bmod N$.

REMARK 3. Depending on the memory capacity of the client, the final exponentiation performed by the client in **Step 5.** may be done by multivariate exponentiation with multiplication complexity $2 \cdot \log_2(U) + 2^n + O(1)$ and memory complexity 2^n or by independent single exponentiation with multiplication complexity $2 \cdot n \cdot \log_2(U) + n + O(1)$ and memory complexity n. Obviously, many tradeoffs are possible, but we consider only these two settings in the following for efficiency comparison.

4.2. Previous Cryptanalysis of RSA-S1.

In this section, we briefly recall the two main attacks known on the RSA-S1 Protocol (more details are given in [10]).

4.2.1. Meet-in-the-Middle Attack.
Since $m^{e(d_1 r_1 + \cdots d_n r_n)} = m^{ed} = m \bmod N$ for all $m \in \mathbb{Z}_N^*$, a *meet-in-the-middle* attack against RSA-S1 consists, assuming for simplicity n even, in computing and storing in a hash table

$$m^{e(d_1 r_1 + \cdots d_{n/2} r_{n/2})} \bmod N$$

for all $(d_1, \dots, d_{n/2}) \in \{0, \dots, U\}^{n/2}$ and searching

$$m^{1 - e(d_{n/2+1} r_{n/2+1} + \cdots d_n r_n)}$$

for $(d_{n/2+1}, \dots, d_n) \in \{0, \dots, U\}^{n/2}$ in this hash table. This simple approach allows to recover the secret exponent d in time complexity $O(U^{n/2})$ and memory complexity $O(U^{n/2})$.

4.2.2. Lattice-based Attack.
Nguyen and Shparlinski [10] proposed a lattice-based attack on the (one-round) RSA-S1 protocol when a small public exponent $e \simeq N^\alpha$ is used, for some $\alpha \geqslant 0$. They showed that one can recover the secret d as long as the sizes of each unknown d_i, for $i \in \{1, \dots, n\}$ satisfies:

$$U < N^{\frac{1-2\alpha}{2(n+1)}}.$$

In particular, their attack can be successful only for $\alpha < 1/2$.

4.3. New Attack against RSA-S1 for Large e

In this section we propose a k-rounds lattice attack (with $k \geqslant 2$) on the RSA-S1 Server-Aided protocol which works for any public exponent e.

When the RSA-S1 Server-Aided protocol is performed in k-rounds, the passive adversary knows k decompositions of the secret d (namely, we have $d = \sum_{i=1}^{n} r_i^{(j)} d_i^{(j)} \bmod (p-1)(q-1)$, for $j \in \{1, \dots, k\}$ where the integers $r_i^{(j)}$ are known to the server and the integers $d_i^{(j)}$ are kept secret by the client and $d_i^{(j)} < U$, for some integer U). The goal is to recover the integers $d_i^{(j)}$, for $i \in \{1, \dots, n\}$ and $j \in \{1, \dots, k\}$. We suppose that $e \simeq N^\alpha$, $r_i^{(j)} \simeq N$ and that p and q have the same size ($p, q \simeq N^{\frac{1}{2}}$). Then for $j \in \{2, \dots, k\}$, we have:

$$\sum_{i=1}^{n} r_i^{(j)} d_i^{(j)} - \sum_{i=1}^{n} r_i^{(1)} d_i^{(1)} = k_j (p-1)(q-1),$$

for some integer $k_j \simeq U$. Then for $j \in \{2, \dots, k\}$, we obtain a polynomial

$$f_j(z_{1,1}, \dots, z_{1,n}, \dots, z_{k,1}, \dots, z_{k,n}, z_2, \dots, z_k)$$

having as root

$$X = (d_1^{(1)}, \dots, d_n^{(1)}, \dots, d_1^{(k)}, \dots, d_n^{(k)}, k_2(p+q-1), \dots, k_k(p+q-1))$$

modulo N with:

$$f_j = z_j + \sum_{i=1}^{n} r_i^{(j)} z_{j,i} - \sum_{i=1}^{n} r_i^{(1)} z_{1,i} \bmod N \qquad (1)$$

We put $f'_j = f_j - z_j$ for $j \in \{2, \dots, k\}$, $K = \lfloor N^{1/2} \rfloor$ and $\omega = nk + k - 1$,

$$\mathcal{M}_1 = \{z_{1,1}, \dots, z_{1,n}, \dots, z_{k,1}, \dots, z_{k,n}\} = \{m_1, \dots, m_{\omega_1}\}$$

with $\omega_1 = \sharp \mathcal{M}_1 = nk$ and $m_1 = z_{1,1}, \dots, m_n = z_{1,n}, \dots, m_{\omega_1} = z_{k,n}$. Each polynomial f'_j can be expressed as a vector with respect to the order $<$ on $\mathcal{M}_1$ (with $m_i < m_{i+1}$). We define the ω-dimensional lattice $\mathcal{L}$ spanned by the rows of the following matrix:

$$\mathcal{A} = \left(\begin{array}{cccc|ccc} K & & & & f'_2 & \cdots & f'_k \\ & K & & & \downarrow & \downarrow & \downarrow \\ & & \ddots & & & & \\ & & & K & & & \\ \hline & & 0 & & N & & \\ & & & & & \ddots & \\ & & & & & & N \end{array}\right) \begin{array}{c} m_1 \\ m_2 \\ \vdots \\ m_{\omega_1} \end{array}$$

The right-hand side is formed by all vectors coming from the set of polynomials $\{f'_2, \dots, f'_k\}$. We have $\det(\mathcal{L}) = K^{nk} N^{k-1}$. The lattice $\mathcal{L}$ contains the vector

$$v = (Kd_1^{(1)}, \dots, Kd_n^{(1)}, \dots, Kd_1^{(k)}, \dots, Kd_n^{(k)}, -k_2(p+q-1), \dots, -k_k(p+q-1))$$

with the Euclidean norm $\|v\| = O(\sqrt{\omega} UK)$. If we can find find the vector v, then we will be able to recover the unknowns $d_i^{(j)}$ for $j \in \{1, \dots, k\}$ and $i \in \{1, \dots, n\}$. By Lemma 2.1 and our heuristic from Section 2.1, the

n	k	δ_{theo}	δ_{exp}	dimension	LLL time(s)
2	2	0.1	0.098	5	0.00159
2	3	0.125	0.123	8	0.01282
2	4	0.136	0.134	11	0.04001
3	2	0.071	0.07	7	0.00303
3	3	0.09	0.088	11	0.02870
3	4	0.1	0.098	15	0.09851

TABLE 2. Experimental Results (RSA-S1) – Average running times (in seconds) of the LLL algorithm and the Gröbner basis computation

vector v is likely to be the shortest vector in the lattice $\mathcal{L}$ if the following condition holds:

$$\|v\| \leqslant \sqrt{\omega}\det(\mathcal{L})^{\omega},$$

which is equivalent to $U \leqslant N^{\frac{k-1}{2(nk+k-1)}}$.

Therefore, we can recover the secret d as long as the sizes of each unknown $d_i^{(j)}$ for $j \in \{1,\ldots,k\}$ and $i \in \{1,\ldots,n\}$ satisfies:

$$U < N^{\frac{k-1}{2(nk+k-1)}} \underset{k\to\infty}{\to} N^{\frac{1}{2(n+1)}}.$$

REMARK 4. For fixed k and for $\alpha \geqslant 1/2$, our attack is successful while the one of Nguyen and Shparlinski does not work. For fixed k and for $1/4 \leqslant \alpha < 1/2$, our bound is better than the one of Nguyen and Shparlinski. For $0 < \alpha < 1/4$, our asymptotic bound is better than the bound of Nguyen and Shparlinski. And for $\alpha = 0$, our asymptotic bound is the same as the bound of Nguyen and Shparlinski.

4.3.1. Experimental Results

The table below lists the theoretical bound $\delta_{\text{theo}} = \frac{k-1}{2(nk+k-1)}$ $(U < N^{\delta})$ and the experimental bound δ_{exp} for a 2048-bit modulus N and for $e \simeq N^{1/2}$ with a few number of rounds $(2 \leqslant k \leqslant 4)$ and a few number of unknown blocks in each decomposition of the secret d $(n \in \{2,3\})$. We run 2^7 experiments for each choice of parameters and Table 2 gives the average running times (in seconds) of the LLL algorithm (using the same configuration as above).

5. ANALYSIS OF A VARIANT OF RSA-S1 SERVER-AIDED PROTOCOL

In this section, we describe and analyze a variant of the server-aided protocol RSA-S1 which might be useful in cases where the client has strong memory constraints and cannot use multi-exponentiation techniques. In this setting, the final exponentiation in the protocol RSA-S1 is simply done by performing independently n multiplications with exponents of size U.

5.1. Description of a New Server-Aided Protocol

Our variant of the RSA-S1 protocol works as follows:

- **Step 1.** The client picks uniformly at random $n \geqslant 1$ "small" integers $d_i \in \mathbb{Z}$, for $i \in \{1,\ldots,n\}$ with $d_i \in \{0,\ldots,U\}$ for some integer U.
- **Step 2.** The client picks uniformly at random n elements $r_i \in \mathbb{Z}_N$, for $i \in \{1,\ldots,n\}$ from the set of vectors satisfying the congruence:

  $$d = r_1{\cdot}d_1 + r_2{\cdot}(d_1d_2) + \cdots + r_n{\cdot}(d_1d_2\ldots d_n) \bmod \varphi(N),$$

- **Step 3.** The client sends the $(n+2)$-tuple $(m, r_1, \ldots, r_n, N)$ to the server.
- **Step 4.** The server computes and sends to the client $z_i = m^{r_i} \bmod N$, for $i \in \{1,\ldots,n\}$.
- **Step 5.**. The client computes

  $$\left(z_1 \cdot \left(z_2 \cdot \left(z_3 \left(z_4 \cdots \left(z_n^{d_n}\right)\cdots\right)^{d_4}\right)^{d_3}\right)^{d_2}\right)^{d_1} \bmod N.$$

  which is equal to $m^d \bmod N$.

In this protocol, the final operation performed by the client is inherently sequential and it is not possible to use multi-exponentiation techniques. The linear random decomposition is replaces by a degree n multi-variate polynomial, one can evaluate using n multiplications using Horner rule. The client final operation consists therefore in n independent single exponentiation (that cannot be parallelized) with multiplication complexity $2 \cdot n \cdot \log_2(U) + n + O(1)$ and memory complexity n. Its efficiency for fixed U is similar to the one of the protocol RSA-S1 but we will see that this protocol (we call RSA-S1H) seems more resistant to cryptanalysis.

It is worth noting that a *meet-in-the-middle* attack against RSA-S1H can be mounted even if the group order is unknown to the adversary (see [32] for instance). It allows to recover the secret exponent d in time complexity $O(U^{n/2})$ and memory complexity $O(U^{n/2})$.

5.2. A One-Round Attack on RSA-S1H Server-Aided Protocol

In this section, we propose a one-round passive attack on RSA-S1H protocol when a small public exponent e is used (for instance $e = 3$).

5.2.1. Description of the Attack

The secret d is decomposed once as $d = r_1d_1 + r_2d_1d_2 + \cdots + r_nd_1d_2\ldots d_n \bmod (p-1)(q-1)$, where the integers $r_1,\ldots,r_n \simeq N$ are known and the integers $d_i \leqslant U$, for $i \in \{1,\ldots,n\}$ are kept secret by the client. We suppose that $e \simeq N^{\alpha}$ with $0 < \alpha < 1/2$ very small and that p and q have the same size $(p, q \simeq N^{\frac{1}{2}})$. In order to recover the secret d, the goal is to recover the integers d_i, for $i \in \{1,\ldots,n\}$. We have

$$e(r_1d_1 + r_2d_1d_2 + \cdots + r_nd_1d_2\ldots d_n) = 1 \bmod (p-1)(q-1),$$

then there exists an integer $k \simeq eU^n$ such that:

$$e(r_1d_1+\cdots+r_nd_1d_2\ldots d_n)+k(p+q-1)-1 = 0 \bmod N.$$

Thus we obtain a linear polynomial $f(x_1,\ldots,x_{n+1})$ having as root

$$X = (d_1, d_1d_2, \ldots, d_1d_2\ldots d_n, k(p+q-1)-1)$$

modulo N with :

$$f = er_1x_1 + \cdots + er_nx_n + x_{n+1} \bmod N.$$

Putting $f = \sum_{i=1}^{n+1} d_ix_i$ (with $d_i = er_i$ for $i \in \{1,\ldots,n\}$ and $d_{n+1} = 1$) and $f = \sum_{i=1}^{n+1} d'_ix_i$ (with $d'_i = d_iU^{i-n} \bmod N$ for $i \in \{1,\ldots,n\}$ and $d'_{n+1} = 1$), one can easily verify that f' has as root

$$X' = (U^{n-1}d_1, U^{n-2}d_1d_2, \ldots, U^1d_1d_2\ldots d_{n-1}, d_1d_2\ldots d_n, k(p+q-1)-1)$$

modulo N. We define the $n+1$-dimensional lattice $\mathcal{L}$ spanned by the rows of the following matrix:

$$\mathcal{A} = \begin{pmatrix} N & 0 & 0 & \cdots & 0 \\ d'_1 & eK & 0 & \cdots & 0 \\ d'_2 & 0 & eK & 0\cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots 0 \\ d'_n & 0 & \cdots & 0 & eK \end{pmatrix}$$

with $K = \lfloor N^{1/2} \rfloor$. We have $\det(\mathcal{L}) = Ne^nK^n$ and the lattice $\mathcal{L}$ contains the vector

$$v = (-k(p+q-1)+1, eKU^{n-1}d_1, eKU^{n-2}d_1d_2, \ldots, eKUd_1d_2\ldots d_{n-1}, eKd_1d_2\ldots d_n)$$

with the Euclidean norm $\|v\| = O(\sqrt{n+1}U^nK)$. If we can find find the vector v, then we will be able to recover the unknowns d_i for $i \in \{1,\ldots,n\}$. By Lemma 2.1 and the Gaussian heuristic, the vector v is likely to be the shortest vector in the lattice $\mathcal{L}$ if the following condition holds:

$$\|v\| \leqslant \sqrt{n+1}\det(\mathcal{L})^{n+1},$$

which is equivalent to $U \leqslant N^{\frac{1-2\alpha}{2n(n+1)}}$.

Therefore, we can recover the secret d as long as the sizes of each unknown d_i for $i \in \{1,\ldots,n\}$ satisfies:

$$U < N^{\frac{1-2\alpha}{2n(n+1)}}.$$

If e is very small (namely $\alpha \simeq 0$), we can thus heuristically recover the secret d as long as the sizes of each unknown d_i for $i \in \{1,\ldots,n\}$ satisfies:

$$U < N^{\frac{1}{2n(n+1)}}.$$

Furthermore, from $\theta = -k(p+q-1)+1$, and the d_i's for $i \in \{1,\ldots,n\}$, one may compute k as $k = \frac{ed^\star-\theta}{N}$, where $d^\star = r_1d_1 + r_2d_1d_2 + \cdots + r_nd_1d_2\ldots d_n$ and then compute $\varphi(N)$ and $p+q-1$ as $\varphi(N) = \frac{ed^\star-1}{k}$ and $p+q-1 = \frac{\theta-1}{-k}$ and yield the factorization of N in polynomial time.

5.2.2. Efficiency comparisons.
Table 3 gives for a 2048-bit and a 3072-bit RSA modulus and for a very small public exponent e ($\alpha \simeq 0$), the theoretical lower bound $n_{theo} = \lfloor \frac{b}{2(n+1)} \rfloor$ (resp. $n_{theo} = \lfloor \frac{b}{2n(n+1)} \rfloor$) on the number of bits of each secret d_i, $i \in \{1,\ldots,n\}$ in the decomposition of the secret d in the RSA-S1 protocol with linear polynomial (resp. RSA-S1H) to prevent our lattice-based attacks (for $n \leqslant 10$).

To prevent *meet-in-the-middle attacks*, it is necessary to use parameters U and n such that $U^{n/2} \geqslant 2^\kappa$ where κ is the security parameter. For instance, to achieve 128-bits security with $\log_2(N) = 3072$, one can use for $n = 4$ a value $U \simeq 2^{76} = 2^{\max(76,256/4)}$ for RSA-S1H (instead of $U \simeq 2^{256}$ for RSA-S1) and for $n = 10$ a value $U \simeq 2^{25.6} = 2^{\max(13,256/10)}$ for RSA-S1H (instead of $U \simeq 2^{139}$ for RSA-S1).

5.2.3. Experimental Results.
The table below lists the theoretical bound $\delta_{\text{theo}} = \frac{1}{2n(n+1)}$ ($U < N^\delta$) and the experimental bound δ_{exp} for a 1024-bit modulus N and for $e = 3$ a few number of unknown blocks in each decomposition of the secret d ($n \in \{2,3,4\}$). We run 2^7 experiments for each choice of parameters and Table 4 gives the average running times (in seconds) of the LLL algorithm (using the same configuration as above).

5.3. A k-Round Attack on RSA-S1H Server-Aided Protocol

In this section, we propose a k-round attack on RSA-S1H protocol which works for any public exponent e.

5.3.1. Description of the Attack
The secret d is decomposed $k \geqslant 2$ times as $d = r_1^{(j)}d_1^{(j)} + r_2^{(j)}d_1^{(j)}d_2^{(j)} + \cdots + r_n^{(j)}d_1^{(j)}d_2^{(j)}\ldots d_n^{(j)} \bmod (p-1)(q-1)$ for $j \in \{1,\ldots,k\}$, where the integers $r_1^{(j)},\ldots,r_n^{(j)} \simeq N$ are known and the integers $d_i^{(j)} \leqslant U$, for $i \in \{1,\ldots,n\}$ are kept secret by the client. For $j \in \{1,\ldots,k\}$, we put $P_j(x_1,\ldots,x_n) = \sum_{i=1}^n r_i^{(j)} \prod_{t=1}^i x_t$ and we suppose that $e \simeq N^\alpha$ with $0 < \alpha$. In order to recover the secret d, the goal is to recover the integers $d_i^{(j)}$, for some $j \in \{1,\ldots,n\}$ and for all $i \in \{1,\ldots,n\}$. For $j \in \{2,\ldots,k\}$, we have

$$P_1(d_1^{(1)},\ldots,d_n^{(1)}) - P_j(d_1^{(j)},\ldots,d_n^{(j)}) = 0 \bmod (p-1)(q-1),$$

then there exists an integer $k^{(j)} \simeq U^n$ such that:

$$P_1(d_1^{(1)},\ldots,d_n^{(1)}) - P_j(d_1^{(j)},\ldots,d_n^{(j)}) + k^{(j)}(p+q-1) = 0 \bmod N.$$

For each $j \in \{2,\ldots,k\}$, we thus obtain a linear multivariate polynomial over $\mathbb{Z}_N$

$$f_j(z_{1,1},\ldots,z_{1,n},\ldots,z_{k,1},\ldots,z_{k,n},z_2,\ldots,z_k)$$

having as root the $(nk+k-1)$-tuple

$$X = (d_1^{(1)},\ldots,d_1^{(1)}\ldots d_n^{(1)},\ldots,d_1^{(k)},\ldots,d_1^{(k)}\ldots d_n^{(k)}, k^{(2)}s,\ldots,k^{(k)}s)$$

	$\log_2(N)$	1	2	3	4	5	6	7	8	9	10
RSA-S1	2048	512	341	256	204	170	146	128	113	102	93
RSA-S1H	2048	512	170	85	51	34	24	18	14	11	9
RSA-S1	3072	768	512	384	307	256	219	192	170	153	139
RSA-S1H	3072	768	256	128	76	51	36	27	21	17	13

TABLE 3. Efficiency comparisons (RSA-S1 and RSA-S1H)

n	δ_{theo}	δ_{exp}	dimension	LLL time(s)
2	0.083	0.082	3	0.002205
3	0.0416	0.0414	4	0.001315
4	0.025	0.024	5	0.001572

TABLE 4. Experimental Results (RSA-S1H) – Average running times (in seconds) of the LLL algorithm and the Gröbner basis computation

modulo N with $s = p + q - 1$ and :

$$f_j = z_j + \sum_{i=1}^{n} r_i^{(1)} z_{1,i} - \sum_{i=1}^{n} r_i^{(j)} z_{j,i} \bmod N$$

Putting

$$g_j = z_j + \sum_{i=1}^{n} r_i^{(1)} U^{i-n} z_{1,i} - \sum_{i=1}^{n} r_i^{(j)} U^{i-n} z_{j,i} \bmod N,$$

one can easily verify that g_j has as root

$$X' = (U^{n-1}d_1^{(1)}, \ldots, d_1^{(1)} \ldots d_n^{(1)}, \ldots, U^{n-1}d_1^{(k)}, \ldots, d_1^{(k)} \ldots d_n^{(k)}, k^{(2)}s, \ldots, k^{(k)}s)$$

modulo N. We put $f'_j = g_j - z_j$ for $j \in \{2, \ldots, k\}$, $K = \lfloor N^{1/2} \rfloor$ and $\omega = nk + k - 1$, $\mathcal{M}_1 = \{z_{1,1}, \ldots, z_{1,n}, \ldots, z_{k,1}, \ldots, z_{k,n}\} = \{m_1, \ldots, m_{\omega_1}\}$ with $\omega_1 = \sharp\mathcal{M}_1 = nk$ and $m_1 = z_{1,1}, \ldots, m_n = z_{1,n}, \ldots, m_{\omega_1} = z_{k,n}$. Each polynomial f'_j can be expressed as a vector with respect to the order $<$ on $\mathcal{M}_1$ (with $m_i < m_{i+1}$). Then we define the ω-dimensional lattice $\mathcal{L}$ spanned by the rows of the following matrix:

$$\mathcal{A} = \left(\begin{array}{c|c} \begin{array}{cccc} K & & & \\ & K & & \\ & & \ddots & \\ & & & K \\ \hline & & 0 & \end{array} & \begin{array}{ccc} f'_2 & \cdots & f'_k \\ \downarrow & \downarrow & \downarrow \\ & & \\ & & \\ \hline & & \\ & & \\ & & \end{array} \end{array}\right) \begin{array}{c} m_1 \\ m_2 \\ \vdots \\ m_{\omega_1} \end{array}$$

$$\begin{array}{c} \\ \\ \\ \\ \hline N \\ \\ \\ N \end{array}$$

The right-hand side is formed by all vectors coming from the set of polynomials $\{f'_2, \ldots, f'_k\}$. We have $\det(\mathcal{L}) = K^{nk}N^{k-1}$. The lattice $\mathcal{L}$ contains the vector

$$v = (KU^{n-1}d_1^{(1)}, \ldots, Kd_1^{(1)} \ldots d_n^{(1)}, \ldots, KU^{n-1}d_1^{(k)}, \ldots, Kd_1^{(k)} \ldots d_n^{(k)}, -k^{(2)}s, \ldots, -k^{(k)}s)$$

with the Euclidean norm $\|v\| = O(\sqrt{\omega}U^nK)$. If we can find find the vector v, then we will be able to recover the unknowns $d_i^{(j)}$ for $j \in \{1, \ldots, k\}$ and $i \in \{1, \ldots, n\}$. By Lemma 2.1 and the Gaussian heuristic, the vector v is likely to be the shortest vector in the lattice $\mathcal{L}$ if the following condition holds:

$$\|v\| \leqslant \sqrt{\omega}\det(\mathcal{L})^{\omega},$$

which is equivalent to

$$U \leqslant N^{\frac{k-1}{2n(nk+k-1)}}$$

Therefore, we can recover the secret d as long as the sizes of each unknown $d_i^{(j)}$ for $j \in \{1, \ldots, k\}$ and $i \in \{1, \ldots, n\}$ satisfies:

$$U < N^{\frac{k-1}{2n(nk+k-1)}} \underset{k\to\infty}{\to} N^{\frac{1}{2n(n+1)}}.$$

6. CRYPTANALYSIS OF RSA-S2 PROTOCOL

6.1. Description of RSA-S2

Matsumoto, Kato and Imai [3] also proposed another protocol, known as RSA-S2 which employs the *Chinese Remainder Theorem* to speed up the client's computation.

- **Step 0.** The client computes integers $d_p < p$ and $d_q < q$ such that $d_p = d \bmod p - 1$ and $d_q = d \bmod q - 1$.
- **Step 1.** The client picks uniformly at random $n \geqslant 1$ pairs of "small" integers $(d_i, d'_i) \in \mathbb{Z}^2$, for $i \in \{1, \ldots, n\}$ with $d_i, d'_i \in \{0, \ldots, U\}$ for some integer U.
- **Step 2.** The client picks uniformly at random n elements $r_i \in \{0, \ldots, V\}$ (for some integer V), for $i \in \{1, \ldots, n\}$ from the set of vectors satisfying the congruence:

$$d_1 \cdot r_1 + \cdots + d_n \cdot r_n = d_p \bmod p - 1.$$

$$d'_1 \cdot r_1 + \cdots + d'_n \cdot r_n = d_q \bmod q - 1.$$

- **Step 3.** The client sends the $(n + 2)$-tuple $(m, r_1, \ldots, r_n, N)$ to the server.
- **Step 4.** The server computes and sends to the client $z_i = m^{r_i} \bmod N$, for $i \in \{1, \ldots, n\}$.
- **Step 5.**. The client computes

$$\prod_{i=1}^{n} z_i^{d_i} = z_1^{d_1} \ldots z_n^{d_n} \bmod p$$

and

$$\prod_{i=1}^{n} z_i^{d'_i} = z_1^{d'_1} \dots z_n^{d'_n} \bmod q$$

and uses the Chinese Remainder Theorem to recover[3] $\sigma = m^d \bmod N$ satisfying $\sigma_p = m^{d_p} \bmod p$ and $\sigma_q = m^{d_q} \bmod q$.

6.2. Time/Memory Tradeoff Attack using Multi-Evaluation of Polynomials

Let $P(x) \in \mathbb{Z}_N[x]$ be a polynomial of degree less than $\Delta = 2^d$. The multipoint evaluation problem is the task of evaluating P at Δ distinct points $\alpha_0, \dots, \alpha_{\Delta-1} \in \mathbb{Z}_N$. Using Horner's evaluation rule, it is easy to propose a solution that uses $O(\Delta^2)$ addition and multiplication in $\mathbb{Z}_N$ but it is well-known that there exists an algorithm with quasi-linear complexity $O(\Delta \log^2 \Delta) = \tilde{O}(\Delta)$ operations in $\mathbb{Z}_N$ using a divide-and-conquer approach [33, 34].

The multipoint evaluation of univariate polynomials has found numerous application in cryptanalysis (*e.g.* [35, 36]). In this paragraph, we show that this technique allows to break the RSA-S2 Protocol in time $\tilde{O}(U^{n/2})$.

For simplicity, we assume that n is even (but the attack can readily be extended to odd n). Let m be some random element of $\mathbb{Z}_N^*$. We have $m^{e \cdot d_p} = m \bmod p$ (since d_p is the inverse of e modulo $p-1$). With high probability, we have $m^{e \cdot d_p} \neq m \bmod q$ (for $d_p \neq d_q$). Therefore, we have with overwhelming probability $\gcd(m^{e \cdot d_p} - m, N) = p$. In the setting of RSA-S2, a passive adversary knows integers $r_1, \dots, r_n$ such that

$$d_p = d_1 r_1 + \dots + d_n r_n \bmod (p-1)$$

and $d_i \leqslant U$ for $i \in \{1, \dots, n\}$. A naive idea to factor N is thus to compute $\gcd(m^{e \cdot (d_1 r_1 + \dots + d_n r_n)} - m, N)$ for all vectors $(d_1, \dots, d_n) \in \{0, \dots, U\}^n$.

In order to apply a meet-in-the-middle approach, one may be tempted to first compute $m^{e \cdot (d_1 r_1 + \dots + d_{n/2} r_{n/2})}$ for all vectors $(d_1, \dots, d_{n/2}) \in \{0, \dots, U\}^{n/2}$ in time $\tilde{O}(U^{n/2})$ and then compute $m^{e \cdot (d_{n/2+1} r_{n/2+1} + \dots + d_n r_n)}$ for all vectors of integers $(d_{n/2+1}, \dots, d_n) \in \{0, \dots, U\}^{n/2}$ in time $\tilde{O}(U^{n/2})$. However, in order to compute all possible gcd's, we have to consider all the pairs and the overall time complexity is $\tilde{O}(U^n)$.

Using multi-evaluation of polynomials, one can simply consider the polynomial

$$P(X) = \prod_{(d_1, \dots, d_{n/2}) \in \{0, \dots, U\}^{n/2}} (m^{e \cdot (d_1 r_1 + \dots + d_{n/2} r_{n/2})} X - m)$$

defined over $\mathbb{Z}_N[X]$. The degree of P is $\Delta = U^{n/2}$ and one can compute its evaluation at the Δ points $m^{e \cdot (d_{n/2+1} r_{n/2+1} + \dots + d_n r_n)}$ for $(d_{n/2+1}, \dots, d_n) \in \{0, \dots, U\}^{n/2}$ in time $\tilde{O}(\Delta) = \tilde{O}(U^{n/2})$.

We can then simply compute

$$\gcd(P(m^{e \cdot (d_{n/2+1} r_{n/2+1} + \dots + d_n r_n)}), N)$$

for each vector $(d_{n/2+1}, \dots, d_n) \in \{0, \dots, U\}^{n/2}$ and with high probability we will obtain one non-trivial[4] factor of N.

REMARK 5. As mentioned above, Castelluccia, Mykletun and Tsudik [7] revisited in 2006 the RSA-S2 protocol. They proposed several security parameters for RSA moduli of bit-size 1024, 1536 and 2048. For these parameters the time and memory complexity term $U^{n/2}$ of our attack are only equal to 2^{36}, 2^{40} and 2^{44} (respectively) and these parameters are therefore broken.

6.3. A One-Round Attack using Herrmann-May Technique

In this section, we propose a one-round attack on RSA-S2 protocol using Herrmann-May technique that is a Coppermith's technique to find small solutions of a linear polynomial modulo an unknown prime number, where a multiple of that prime is known. In our attack, we suppose that a very small public exponent e (for instance $e = 3$) is used and that small random elements r_i's are used ($V < p$).

6.3.1. Description of the Attack

The secrets d_p and d_q are decomposed once as $d_p = r_1 d_1 + r_2 d_2 + \dots + r_n d_n \bmod (p-1)$, $d_q = r_1 d'_1 + r_2 d_2 + \dots + r_n d'_n \bmod (q-1)$, where the integers $r_1, \dots, r_n \simeq p$ are known and the integers $d_i, d'_i \leqslant U = N^\gamma$, for $i \in \{1, \dots, n\}$ are kept secret by the client. We suppose that $e \simeq N^\alpha$ with $\alpha \simeq 0$ and that p and q have the same size ($p, q \simeq N^{\frac{1}{2}}$) . The goal is to recover the integers d_i, d'_i, for $i \in \{1, \dots, n\}$. In the following, we show how to recover the integers d_i, for $i \in \{1, \dots, n\}$ and the same technique can be used to recover the others integers d'_i, for $i \in \{1, \dots, n\}$. We have

$$e(r_1 d_1 + r_2 d_2 + \dots + r_n d_n) = 1 \bmod (p-1),$$

then there exists an integer $k \simeq eU$ such that $e(r_1 d_1 + r_2 d_2 + \dots + r_n d_n) - 1 = k(p-1)$. Thus we have:

$$e(r_1 d_1 + r_2 d_2 + \dots + r_n d_n) + k - 1 = 0 \bmod p,$$

where the prime p is unknown, we only know the modulus N which is a multiple of p. We obtain a

[3] Given the (precomputed) values $\alpha_p = q \cdot (q^{-1} \bmod p)$ and $\alpha_q = p \cdot (p^{-1} \bmod q)$, we have $\sigma = \sigma_p \alpha_p + \sigma_q \alpha_q \bmod N$.

[4] If one of this gcd's is actually equal to N for some $(d_{n/2+1}, \dots, d_n) \in \{0, \dots, U\}^{n/2}$ and one can obtain a prime divisor of N by looking (in time $\tilde{O}(U^{n/2})$) for the vector $(d_1, \dots, d_{n/2}) \in \{0, \dots, U\}^{n/2}$ for which $\gcd(m^{e \cdot (d_1 r_1 + \dots + d_n r_n)} - m, N)$ is non-trivial.

linear polynomial $f(x_1, \ldots, x_{n+1})$ having as root $X = (d_1, d_2, \ldots, d_n, k)$ modulo p with :

$$f = er_1x_1 + \cdots + er_nx_n + x_{n+1} - 1.$$

We use the following theorem proved by Herrmann and May in [37] to find the root $X = (d_1, d_2, \ldots, d_n, k)$:

THEOREM 6.1. *Let $\varepsilon > 0$ and let $N = pq$ be a sufficiently large composite integer with a divisor $p \geqslant N^\beta$. Let $g(x_1, \ldots, x_\ell) \in \mathbb{Z}[x_1, \ldots, x_\ell]$ be a monic polynomial in ℓ variables. We can find heuristically all solutions $(y_1, \ldots, y_\ell)$ of the equation $f(x_1, \ldots, x_\ell) = 0 \bmod p$ with $|y_1| \leqslant N^{\gamma_1}, \ldots, |y_\ell| \leqslant N^{\gamma_\ell}$ if*

$$\sum_{i=1}^{\ell} \gamma_i \leqslant 1 - (1-\beta)^{\frac{\ell+1}{\ell}} - (\ell+1)(1 - \sqrt[\ell]{1-\beta})(1-\beta) - \varepsilon.$$

The time and space complexity of the algorithm is polynomial in $\log_2 N$ and $(\frac{c_0}{\varepsilon})^\ell$, where c_0 is Euler's constant.

Applying Theorem 6.1 with $\beta = \frac{1}{2}$ and a very small $\varepsilon > 0$, we can heuristically find the root $X = (d_1, d_2, \ldots, d_n, k)$ with $d_1 \leqslant U = N^\gamma, \ldots, d_n \leqslant U = N^\gamma, |k| \leqslant eU = N^{\gamma+\alpha}$ if:

$$(n+1)\gamma + \alpha < 1 - 2^{-\frac{n+2}{n+1}} - \frac{n+2}{2}(1 - 2^{-\frac{1}{n+1}}) - \varepsilon,$$

Therefore, if N sufficiently large and e is very small (namely $\alpha \simeq 0$), for a very small $\varepsilon > 0$ we can heuristically recover the secret d_i, d_i', for $i \in \{1, \ldots, n\}$ as long as the sizes of each unknown d_i, d_i' for $i \in \{1, \ldots, n\}$ satisfies:

$$U < N^{\frac{1 - 2^{-\frac{n+2}{n+1}} - \frac{n+2}{2}(1 - 2^{-\frac{1}{n+1}}) - \varepsilon}{(n+1)}}.$$

REMARK 6.

- We can find the roots in polynomial time whenever $n + 1 \leqslant O(\log_2(\log_2 N))$.
- In the proof of Theorem 6.1, Herrmann and May used Coppermith's method with the following collection of polynomials which share a common root modulo p^t:

  $$g_{i_2,\ldots,i_\ell,j} = x_2^{i_2} \ldots x_\ell^{i_\ell} g^j N^{\max(t-j,0)},$$

  where $i_2, \ldots, i_\ell \in \{0, \ldots, m\}$ such that $\sum_{b=2}^{n} i_b + j \leqslant m$, g is monic in x_1, $m = \lceil \frac{\ell(\frac{1}{\pi}(1-\beta)^{-0.278465} - \beta \ln(1-\beta))}{\varepsilon} \rceil$ and $t = \lceil (1 - \sqrt[\ell]{1-\beta})m \rceil$.
- In the previous theorem, the bounds of the unknowns d_i, d_i', for $i \in \{1, \ldots, n\}$ are identical and there is no condition on the bound U of the unknowns. In particular, the results from [38] cannot be applied in this setting.

6.3.2. Experimental Results

The table below lists the theoretical bound $\delta_{\text{theo}} = \frac{1 - 2^{-\frac{n+2}{n+1}} - \frac{n+2}{2}(1 - 2^{-\frac{1}{n+1}}) - \varepsilon}{(n+1)}$ ($U < N^{\delta_{\text{theo}}}$) and the experimental bound δ_{exp} for a 1024-bit modulus $N = pq$ and $p, q \simeq N^{1/2}$, for $e = 3$ and with a few number of unknown blocks in each decomposition of the secrets d_p and d_q ($n \leqslant 5$). In our experiments, we obtained after the LLL reduction and the Gröbner basis computations, a system of polynomials of dimension 1 but were always able to find the desired root. We run 2^4 experiments for each choice of parameters and Table 5 gives the average running times (in seconds) of the LLL algorithm and the Gröbner basis computation (using the same configuration as above).

REMARK 7. Castelluccia *et al.* proposed parameters in [7] for $n = 2$. Even if one doubles the bit-size of the proposed U in order to prevent the meet-in-the-middle attack from the previous section, we obtain for RSA moduli of bit-size 1024, 1536 and 2048, U equal to 2^{72}, 2^{80} and 2^{88} , i.e. $U \simeq N^{0.070}$, $U \simeq N^{0.052}$ an $U \simeq N^{0.044}$ (respectively). Our attack works for $U \leqslant N^{0.0635}$ and the last two (updated) parameters are therefore broken in practice.

6.4. A Two-Round Attack using Herrmann-May technique

In this section, we give a description of a Two-Round attack on RSA-S2 protocol using Herrmann-May technique when an arbitrary public exponent e is used. The secrets d_p and d_q are decomposed twice as $d_p = r_1^1 d_1^1 + r_2^1 d_2^1 + \cdots + r_n^1 d_n^1 \bmod (p-1)$, $d_p = r_1^2 d_1^2 + \cdots + r_n^2 d_n^2 \bmod (p-1)$, $d_q = r_1^1 c_1^1 + r_2^1 c_2^1 + \cdots + r_n^1 c_n^1 \bmod (q-1)$ and $d_q = r_1^2 c_1^2 + \cdots + r_n^2 c_n^2 \bmod (q-1)$, where the integers $r_i^1, r_i^2 \simeq p$ for $i \in \{1, \ldots, n\}$ are known and the integers $d_i^1, d_i^2, c_i^1, c_i^2 \leqslant U = N^\gamma$, for $i \in \{1, \ldots, n\}$ are kept secret by the client. We suppose that $e \simeq N^\alpha$ with $\alpha > 0$ arbitrary and that p and q have the same size ($p, q \simeq N^{\frac{1}{2}}$) . The goal is to recover the integers $d_i^1, d_i^2, c_i^1, c_i^2$, for $i \in \{1, \ldots, n\}$. We have

$$\sum_{i=1}^{n} (r_i^2 d_i^2 - r_i^1 d_i^1) = 0 \bmod (p-1),$$

then there exists an integer $k \simeq U$ such that

$$\sum_{i=1}^{n} (r_i^2 d_i^2 - r_i^1 d_i^1) + k = 0 \bmod p,$$

where the prime p is unknown. If we proceed as in the previous section, for a very small $\varepsilon > 0$, we can heuristically find $X = (d_1^1, \ldots, d_n^1, d_1^2, \ldots, d_n^2, k)$ if:

$$U < N^{\frac{1 - 2^{-\frac{2n+2}{2n+1}} - \frac{2n+2}{2}(1 - 2^{-\frac{1}{2n+1}}) - \varepsilon}{(2n+1)}}.$$

If we use the same technique with the decompositions of d_q, then for an arbitrary e we can heuristically

n	ε	m	δ_{theo}	δ_{exp}	dimension	LLL time(s)	Gröbner basis time(s)
1	0.010	8	0.098	0.107	45	7.06	0.047
1	0.007	12	0.100	0.110	91	164.68	0.321
2	0.020	6	0.057	0.047	84	69.61	15.113
2	0.017	7	0.058	0.051	120	383.53	58.107
3	0.040	4	0.035	0.017	70	29.46	15.79
3	0.035	5	0.037	0.018	126	422.67	174.97

TABLE 5. Experimental Results (RSA-S2) – Average running times (in seconds) of the LLL algorithm and the Gröbner basis computation

recover the secret d as long as the sizes of each unknown $d_i^1, d_i^2, c_i^1, c_i^2$, for $i \in \{1, \ldots, n\}$ satisfies:

$$U < N^{\frac{1-2^{-\frac{2n+2}{2n+1}} - \frac{2n+2}{2}(1-2^{-\frac{1}{2n+1}})-\varepsilon}{(2n+1)}}.$$

7. EFFICIENCY OF SERVER-AIDED RSA PROTOCOLS

In this section, we compare the efficiency of the protocols studied in the previous sections. We consider 3072-bit RSA moduli and different values for $n \in \{1, 2, 4, 8\}$. Since our two families of attacks (for small and arbitrary public exponents), succeed asymptotically (when k goes to ∞) for the same bound U, we do not distinguish them in the comparison. For RSA-S1 and RSA-S2, we distinguish two settings depending on whether the client has enough memory to perform multi-exponentiation in **Step 5.** of these two protocols.

Acknowledgments. The authors are supported in part by the French ANR ALAMBIC Project (ANR-16-CE39-0006).

REFERENCES

[1] Hohenberger, S. and Lysyanskaya, A. (2005) How to securely outsource cryptographic computations. In Kilian, J. (ed.), *TCC 2005*, February, LNCS, **3378**, pp. 264–282. Springer, Heidelberg.

[2] Chevalier, C., Laguillaumie, F., and Vergnaud, D. (2016) Privately outsourcing exponentiation to a single server: Cryptanalysis and optimal constructions. In Askoxylakis, I. G., Ioannidis, S., Katsikas, S. K., and Meadows, C. A. (eds.), *ESORICS 2016, Part I*, September, LNCS, **9878**, pp. 261–278. Springer, Heidelberg.

[3] Matsumoto, T., Kato, K., and Imai, H. (1990) Speeding up secret computations with insecure auxiliary devices. In Goldwasser, S. (ed.), *CRYPTO'88*, August, LNCS, **403**, pp. 497–506. Springer, Heidelberg.

[4] Laih, C.-S., Yen, S.-M., and Harn, L. (1993) Two efficient server-aided secret computation protocols based on the addition sequence. In Imai, H., Rivest, R. L., and Matsumoto, T. (eds.), *ASIACRYPT'91*, November, LNCS, **739**, pp. 450–459. Springer, Heidelberg.

[5] Béguin, P. and Quisquater, J.-J. (1995) Fast server-aided RSA signatures secure against active attacks. In Coppersmith, D. (ed.), *CRYPTO'95*, August, LNCS, **963**, pp. 57–69. Springer, Heidelberg.

[6] Lim, C. H. and Lee, P. J. (1995) Security and performance of server-aided RSA computation protocols. In Coppersmith, D. (ed.), *CRYPTO'95*, August, LNCS, **963**, pp. 70–83. Springer, Heidelberg.

[7] Castelluccia, C., Mykletun, E., and Tsudik, G. (2006) Improving secure server performance by re-balancing SSL/TLS handshakes. In Lin, F.-C., Lee, D.-T., Lin, B.-S., Shieh, S., and Jajodia, S. (eds.), *ASIACCS 06*, March, pp. 26–34. ACM Press.

[8] Chen, X., Li, J., Ma, J., Tang, Q., and Lou, W. (2012) New algorithms for secure outsourcing of modular exponentiations. In Foresti, S., Yung, M., and Martinelli, F. (eds.), *ESORICS 2012*, September, LNCS, **7459**, pp. 541–556. Springer, Heidelberg.

[9] Wang, Y., Wu, Q., Wong, D. S., Qin, B., Chow, S. S. M., Liu, Z., and Tan, X. (2014) Securely outsourcing exponentiations with single untrusted program for cloud storage. In Kutylowski, M. and Vaidya, J. (eds.), *ESORICS 2014, Part I*, September, LNCS, **8712**, pp. 326–343. Springer, Heidelberg.

[10] Nguyen, P. Q. and Shparlinski, I. (2001) On the insecurity of a server-aided RSA protocol. In Boyd, C. (ed.), *ASIACRYPT 2001*, December, LNCS, **2248**, pp. 21–35. Springer, Heidelberg.

[11] Merkle, J. (2000) Multi-round passive attacks on server-aided RSA protocols. In Jajodia, S. and Samarati, P. (eds.), *ACM CCS 00*, November, pp. 102–107. ACM Press.

[12] Pfitzmann, B. and Waidner, M. (1993) Attacks on protocols for server-aided RSA computation. In Rueppel, R. A. (ed.), *EUROCRYPT'92*, May, LNCS, **658**, pp. 153–162. Springer, Heidelberg.

[13] Jakobsson, M. and Wetzel, S. (2001) Secure server-aided signature generation. In Kim, K. (ed.), *PKC 2001*, February, LNCS, **1992**, pp. 383–401. Springer, Heidelberg.

[14] Merkle, J. and Werchner, R. (1998) On the security of server-aided RSA protocols. In Imai, H. and Zheng, Y. (eds.), *PKC'98*, February, LNCS, **1431**, pp. 99–116. Springer, Heidelberg.

[15] Joye, M. and Lepoint, T. (2012) Partial key exposure on rsa with private exponets larger than n. *ISPEC 2012*, Lecture Notes in Computer Science, **7232**. Springer.

[16] Aono, Y. (2009) A new lattice construction for partial key exposure attack for RSA. In Jarecki, S. and Tsudik, G. (eds.), *PKC 2009*, March, LNCS, **5443**, pp. 34–53. Springer, Heidelberg.

Protocol	n	$\log_2(U)$	Communication	Computation	Memory
RSA-S1 w. multi-exp	1	768	4	1152	1
RSA-S1 w/o multi-exp.				1152	1
RSA-S1H				1152	1
RSA-S1 w. multi-exp	2	512	6	897	3
RSA-S1 w/o multi-exp.		512		1536	2
RSA-S1H		256		768	2
RSA-S1 w. multi-exp	4	307	10	606	15
RSA-S1 w/o multi-exp.		307		1842	4
RSA-S1H		76		456	4
RSA-S1 w. multi-exp	8	170	18	587	256
RSA-S1 w/o multi-exp.		170		2040	8
RSA-S1H		32		384	8
RSA-S2 w. multi-exp	2	128	5	115	2.5
RSA-S2 w/o multi-exp.		128		192	2
RSA-S2 w. multi-exp	4	64	8	75	11.5
RSA-S2 w/o multi-exp.		64		192	4

TABLE 6. Efficiency Comparison of Server-Aided Protocols for 128-bit Security with 3078-bit RSA Moduli. The communication and memory complexity are given in 3078-bit integers and the computational complexity is given in modular multiplications of 3072-bits integers (assuming multiplications of 1536-bits integers for is 1/4 of this cost).

[17] Blömer, J. and May, A. (2003) New partial key exposure attacks on RSA. In Boneh, D. (ed.), *CRYPTO 2003*, August, LNCS, **2729**, pp. 27–43. Springer, Heidelberg.

[18] Ernst, M., Jochemsz, E., May, A., and de Weger, B. (2005) Partial key exposure attacks on RSA up to full size exponents. In Cramer, R. (ed.), *EUROCRYPT 2005*, May, LNCS, **3494**, pp. 371–386. Springer, Heidelberg.

[19] Boneh, D. and Durfee, G. (1999) Cryptanalysis of RSA with private key d less than $N^{0.292}$. In Stern, J. (ed.), *EUROCRYPT'99*, May, LNCS, **1592**, pp. 1–11. Springer, Heidelberg.

[20] Minkowski, H. (1910) *Geometrie der Zahlen*.

[21] Lenstra, A. K., Lenstra, H. W. J., and Lovász, L. (1982) Factoring polynomials with rational coefficients. *Math. Ann.*, **261**, 515–534.

[22] Coppersmith, D. (1996) Finding a small root of a univariate modular equation. In Maurer, U. M. (ed.), *EUROCRYPT'96*, May, LNCS, **1070**, pp. 155–165. Springer, Heidelberg.

[23] Coppersmith, D. (1996) Finding a small root of a bivariate integer equation; factoring with high bits known. In Maurer, U. M. (ed.), *EUROCRYPT'96*, May, LNCS, **1070**, pp. 178–189. Springer, Heidelberg.

[24] Jochemsz, E. and May, A. (2006) A strategy for finding roots of multivariate polynomials with new applications in attacking RSA variants. In Lai, X. and Chen, K. (eds.), *ASIACRYPT 2006*, December, LNCS, **4284**, pp. 267–282. Springer, Heidelberg.

[25] Howgrave-Graham, N. (1997) Finding small roots of univariate modular equations revisited. In Darnell, M. (ed.), *6th IMA International Conference on Cryptography and Coding*, December, LNCS, **1355**, pp. 131–142. Springer, Heidelberg.

[26] Möller, B. (2001) Algorithms for multi-exponentiation. In Vaudenay, S. and Youssef, A. M. (eds.), *SAC 2001*, August, LNCS, **2259**, pp. 165–180. Springer, Heidelberg.

[27] Avanzi, R. M. (2005) The complexity of certain multi-exponentiation techniques in cryptography. *Journal of Cryptology*, **18**, 357–373.

[28] Takayasu, A. and Kunihiro, N. (2014) Partial key exposure attacks on RSA: Achieving the boneh-durfee bound. In Joux, A. and Youssef, A. M. (eds.), *SAC 2014*, August, LNCS, **8781**, pp. 345–362. Springer, Heidelberg.

[29] Takayasu, A. and Kunihiro, N. (2017) A tool kit for partial key exposure attacks on RSA. In Handschuh, H. (ed.), *CT-RSA 2017*, February, LNCS, **10159**, pp. 58–73. Springer, Heidelberg.

[30] Herrmann, M. and May, A. (2010) Maximizing small root bounds by linearization and applications to small secret exponent RSA. In Nguyen, P. Q. and Pointcheval, D. (eds.), *PKC 2010*, May, LNCS, **6056**, pp. 53–69. Springer, Heidelberg.

[31] Peng, L., Hu, L., Lu, Y., Xu, J., and Huang, Z. (2017) Cryptanalysis of dual rsa. *Designs, Codes and Cryptography*, **83**, 1–21.

[32] Coron, J.-S., Lefranc, D., and Poupard, G. (2005) A new baby-step giant-step algorithm and some applications to cryptanalysis. In Rao, J. R. and Sunar, B. (eds.), *CHES 2005*, August / September, LNCS, **3659**, pp. 47–60. Springer, Heidelberg.

[33] Bostan, A., Gaudry, P., and Schost, É. (2007) Linear recurrences with polynomial coefficients and application to integer factorization and Cartier-Manin operator. *SIAM J. Comput.*, **36**, 1777–1806.

[34] Fiduccia, C. M. (1972) Polynomial evaluation via the division algorithm: The Fast Fourier Transform revisited. In Fischer, P. C., Zeiger, H. P., Ullman, J. D., and Rosenberg, A. L. (eds.), *4th Annual ACM Symposium on Theory of Computing*, pp. 88–93. ACM.

[35] Coron, J.-S., Joux, A., Mandal, A., Naccache, D., and Tibouchi, M. (2011) Cryptanalysis of the RSA subgroup assumption from TCC 2005. In Catalano, D., Fazio, N., Gennaro, R., and Nicolosi, A. (eds.), *PKC 2011*, March, LNCS, **6571**, pp. 147–155. Springer,

Heidelberg.

[36] Chen, Y. and Nguyen, P. Q. (2012) Faster algorithms for approximate common divisors: Breaking fully-homomorphic-encryption challenges over the integers. In Pointcheval, D. and Johansson, T. (eds.), *EUROCRYPT 2012*, April, LNCS, **7237**, pp. 502–519. Springer, Heidelberg.

[37] Herrmann, M. and May, A. (2008) Solving linear equations modulo divisors: On factoring given any bits. In Pieprzyk, J. (ed.), *ASIACRYPT 2008*, December, LNCS, **5350**, pp. 406–424. Springer, Heidelberg.

[38] Takayasu, A. and Kunihiro, N. (2014) Better lattice constructions for solving multivariate linear equations modulo unknown divisors. *IEICE Trans. Fundamentals*, **97**, 1259–1272.