Cryptanalysis of Modular Exponentiation Outsourcing Protocols

Charles Bouillaguet[1], Florette Martinez[1] and Damien Vergnaud[2]

[1]Sorbonne Université, CNRS, LIP6, F-75005 Paris, France
[2]Sorbonne Université, CNRS, LIP6, F-75005 Paris, France and Institut Universitaire de France
Email: charles.bouillaguet@lip6.fr, florette.martinez@lip6.fr and damien.vergnaud@lip6.fr

Public-key cryptographic primitives are time-consuming for resource-constrained devices. A classical problem is to securely offload group exponentiations from a (comparatively) weak device — the client — to an untrusted more powerful device — the server. A delegation protocol must usually meet two security objectives: *privacy* – the exponent or the base should not be revealed to a passive adversary — and *verifiability* — a malicious server should not be able to make the client accept an invalid value as the result of the delegated computation. Most proposed protocols relies on a secret splitting of the exponent and the base and a considerable amount of literature has been devoted to their analysis.
Recently, Su, Zhang and Xue [*The Computer Journal, 2020]* and Rangasamy and Kuppusamy *[Indocrypt 2018]* proposed outsourcing protocols for modular exponentiations. They claim that their protocols achieve security (privacy and verifiability). We show that these claims are flawed and that their schemes are broken beyond repair. They remain insecure even if one increases significantly the proposed parameters (and consequently the protocols computational and communication complexities). Our attacks rely on standard lattice-based cryptanalytic techniques, namely the *Coppersmith methods* to find small integer zeroes of modular multivariate polynomials and simultaneous Diophantine approximation methods for the so-called *approximate greatest common divisor problem.*

Keywords: Secure outsourcing; Modular exponentiation; Privacy; Verifiability; Cryptanalysis

1. INTRODUCTION

Group exponentiation is a fundamental operation in public-key cryptography as it is used in RSA-based and discrete-logarithm based protocols. Since the computational resources can be very limited on certain devices, it is natural, as most of the devices are online or directly connected to a powerful device, to consider securely delegating some sensitive and costly exponentiation to an untrusted device capable of carrying out large operations. A delegation protocol must usually meet two security objectives: *privacy* – the exponent or the base should not be revealed to a passive adversary – and *verifiability* – a malicious server should not be able to make the client accept an invalid value as the result of the delegated computation

This paper presents several lattice-based attacks on two group exponentiation outsourcing protocols recently proposed by Su, Zhang and Xue [1] and Rangasamy and Kuppusamy [2]. In both cases, the proposed protocols are simple and efficient and the authors claim that they achieve security (privacy and verifiability). We show that these claims are flawed and that their schemes are broken beyond repair.

1.1. Prior Work on Exponentiation Outsourcing Protocols

The problem of outsourcing cryptographic operations has already received a lot of attention but there has been a recent regain of interest with the development of mobile technologies. In 2005, Hohenberger and Lysyanskaya [3] proposed formal security definitions for securely outsourcing computations from a computationally limited device, called the *client*, to untrusted helpers, called the *servers*. Delegating a cryptographic operation presents many risks since they usually involve sensitive information which should not be revealed to

potential adversaries. Moreover, since the servers are not fully trusted, a delegation protocol should enable clients to verify the correctness of the result returned by the server with high probability. Obviously, to be of practical interest, these delegation protocols must have a computational cost for the client lower than that of the delegated computation.

Hohenberger and Lysyanskaya notably presented an efficient scheme to securely outsource group exponentiation to two, possibly dishonest, servers that are physically separated (and do not communicate). Since this separation of the two servers is actually a strong assumption, recent works focus on outsourcing group exponentiation to a *single* computationally stronger server. It has been a very active research topic in which numerous protocols have been proposed [4, 5, 6, 7, 8, 9, 10]) and many of these proposals were subsequently broken [11, 12, 13, 14, 15]).

In 2018, Rangasamy and Kuppusamy [2] presented a protocol named MExpSOS for outsourcing modular exponentiations to a single, malicious computational resource. Their protocol is presented for delegation of exponentiation modulo a prime number as well as modulo an RSA modulus. It is simple and efficient, and does not require any pre-computation from the client. Rangasamy and Kuppusamy claimed that their scheme achieve the two fundamental security properties, namely privacy of inputs and verifiability of outputs, and claimed that it is the "*best to-date outsourcing scheme for single-server case*". They also proposed another scheme based on similar ideas for simultaneous exponentiations.

In 2020, Su, Zhang and Xue [1] presented another protocol called MCExp for outsourcing exponentiations modulo an RSA modulus to a single, malicious computational resource. It requires pre-computation from the client but relies on similar ideas to those of the MExpSOS protocol. It is also simple and efficient and Su *et al.* claimed that it achieves privacy and verifiability. They also proposed another scheme for outsourcing simultaneous composite modular exponentiations.

1.2. Prior Work on Lattice-Based Cryptanalysis

Some attacks we present in the paper rely on *Coppersmith's methods*, a classical technique in lattice-based cryptanalysis. These methods have been introduced in 1996 by Coppersmith to find small integer zeroes of modular polynomials of one or two variables [16, 17]. Since their introduction, many generalizations of the methods have been proposed to deal with more variables (e.g., [18, 19, 20, 21, 22]) or multiple moduli (e.g., [23, 24]). In this paper, we rely on variants proposed Ernst, Jochemsz, May and de Weger in 2005 [25] for specific trivariate polynomial equations of small degree. The techniques from Ernst *et al.* are currently the best known for solving the particular equations we are studying in this article and any improvement in these techniques would provide an immediate improvement to our attacks.

We also present attacks based on another standard lattice-based cryptanalytic technique from simultaneous Diophantine approximation for the *approximate greatest common divisor problem* [26, 27, 28]). This computational problem is to determine a secret integer p when one is given many samples of the form $x_i = p \cdot q_i + r_i$ for small error terms r_i. A simple approach for solving it relies on a simple lattice-based algorithm due to Lagarias [26] for simultaneous Diophantine approximation. The technique was first proposed by Howgrave-Graham in [27], then expanded in [28].

1.3. Contributions

We prove that the protocols MExpSOS and MCExp as proposed in [2] and [1] are insecure. Both schemes do not achieve the claimed privacy and verifiability security properties (without increasing the size of the parameters to the point of making the delegation protocol more expensive than the modular exponentiation computation itself).

We first underline a major security break in MCExp from a single execution of the protocol, as two of its parameters are too small to resist exhaustive search. This allows us to obtain readily a multiple of the Euler totient function $\varphi(N)$ of the underlying RSA modulus N and then using a classical algorithm due to Rabin [29] to factor N and obtain the (supposedly) secret base and exponent of the delegated exponentiation.

We then consider different ways of fixing the scheme MCExp (by increasing only one of the small parameters, then both of them). Using Coppersmith's methods, we show that even the modified schemes are also broken, from a single execution of the protocol, for a wide range of parameters. For each variant, we present a simple attack that allows to factor N and retrieve the base and exponent of the delegated exponentiation. This information is sufficient to also break the verifiability property of MCExp.

As a final nail in MCExp's coffin, we present an even more devastating attack against the modified protocol when the adversary can passively eavesdrop several runs of the delegation protocol for the same exponent. This is a particularly important use-case for the RSA primitive in which the client may want to delegate the computation of signatures for a fixed secret signing exponent. Our attack relies methods for the approximate greatest common divisor problem. The methods used on this attack can also be used to improve the attack by Mefenza and Vergnaud [15] on a server-aided RSA protocol with additive key splitting in the case where the adversary has access to several delegations. Finally, since MExpSOS is very close to the modified MCExp protocol, we can also apply our attack and show that MExpSOS does not achieve privacy and

verifiability.

The remainder of this paper is organized as follows. Section 2 introduces the notations, the security definitions and describes the MCExp and MExpSOS outsourcing computation schemes. Section 3 reviews the lattice-based cryptanalytic techniques we used in the paper. Our attacks against the MCExp protocol when the client outsources a single exponentiation are described in Section 4. Eventually, Section 5 presents our devastating attacks against MExpSOS and MCExp in the case where the adversary is able to observe more than one exponentiation delegation.

2. PRELIMINARIES

2.1. Notations

All logarithms are base 2. We denote the security parameter by $\lambda \in \mathbb{N}$ which is given as input to all algorithms in unary form 1^λ. Algorithms are randomized unless otherwise stated, and PPT stands for "probabilistic polynomial-time," in the security parameter. We might use the terms "efficient" and PPT interchangeably. We denote random sampling from a finite set X according to the uniform distribution with $x \xleftarrow{\$} X$. We also use the symbol $\xleftarrow{\$}$ for assignments from randomized algorithms, while we denote assignment from deterministic algorithms and calculations with the symbol $\leftarrow$. If n is an integer, we write $\mathbb{Z}_n$ for the ring $\mathbb{Z}/n\mathbb{Z}$, $\mathbb{Z}_n^*$ for the invertible elements of $\mathbb{Z}_n$ and $\varphi(n)$ for the Euler totient function of n (with $\varphi(n) = \#\mathbb{Z}_n^*$). As usual, the notation $\mathrm{poly}(\lambda)$ denotes any polynomial in λ and $f \in \mathrm{negl}(\lambda)$ denotes a function that decreases faster than the inverse of any polynomial in λ; such functions are called *negligible*.

2.2. Exponentiation Delegation: Definitions

In the following, we consider group exponentiation protocols in two different settings, namely public prime-order groups and in secret composite order groups. To simplify the exposition[3] we consider only exponentiation in the multiplicative group of the residue ring $\mathbb{Z}_N$ for some integer $N \geq 2$. These settings are formalized thanks to a PPT algorithm GrpGen which is given as input a security parameter λ and outputs a pair $\mathsf{par} = (N, \varphi(N))$ where $\log(N) = \mathrm{poly}(\lambda)$:

- we say that GrpGen is a *prime-order group generator* if for all $\lambda \in \mathbb{N}$ and all $\mathsf{par} = (N, \varphi(N))$ output by $\mathsf{GrpGen}(1^\lambda)$, N is a prime number (and thus $\varphi(N) = N - 1$);
- we say that GrpGen is an *RSA group generator* if for all $\lambda \in \mathbb{N}$ and all $\mathsf{par} = (N, \varphi(N))$ output by $\mathsf{GrpGen}(1^\lambda)$, $N = pq$ is the product of two distinct odd primes p and q (and thus $\varphi(N) = (p-1)(q-1)$).

[3] It is worth noting that most of our attacks can be applied directly to exponentiation in general groups.

We consider the delegation of an exponentiation as a 2-party PPT interactive protocol between a client $\mathcal{C}$ and a server $\mathcal{S}$. It is natural to assume that $\mathcal{C}$ knows the group $\mathbb{Z}_N^*$ in which it needs to perform an exponentiation and following [2, 1], we also assume that it knows its order $\varphi(N)$. We assume that $\mathcal{S}$ does not know the group $\mathbb{Z}_N^*$ *a priori*; this approach somehow contradicts Kerckhoff's principles [30] but it is adopted in [2]. Anyway, we will show that the schemes from [2, 1] are insecure even with this very strong assumption.

Given a security parameter λ, a pair $\mathsf{par} = (N, \varphi(N))$ output by $\mathsf{GrpGen}(1^\lambda)$, $u \in \mathbb{Z}_N$ and $a \in \mathbb{Z}_{\varphi(N)}$, we denote as $(y_\mathcal{C}, y_\mathcal{S}, tr) \leftarrow (\mathcal{C}(1^\lambda, \mathsf{par}, (a, u)), \mathcal{S}(1^\lambda))$ the protocol at the end of which $\mathcal{C}$ gets $y_\mathcal{C}$ and $\mathcal{S}$ gets $y_\mathcal{S}$ and the string tr denotes the full transcript of the interaction.

In the present paper, the protocols investigated are all of the same form: (1) $\mathcal{C}$ performs some private computation and then sends to $\mathcal{S}$ one or several bases and their corresponding exponents and a modulus (possibly different from N) to the server; (2) $\mathcal{S}$ then performs the required exponentiations (whose bases, exponents and modulus were sent to him by $\mathcal{C}$) and sends back the obtained values to $\mathcal{C}$; (3) $\mathcal{C}$ finalizes the computations thanks to the values sent by $\mathcal{S}$. The protocols only require one round of communication, their transcripts tr thus consist only of the strings sent at the end of steps (1) and (2) and the output of $\mathcal{S}$ is $y_\mathcal{S} = \emptyset$ the empty string.

The correctness requirement for delegation of a exponentiation means that when the server and the client follow honestly the protocol, the client's output is actually the expected exponentiation.

DEFINITION 2.1 (Correctness). *Let λ be a positive integer. We say that $(\mathcal{C}, \mathcal{S})$ satisfies* correctness *if*

$$\Pr\left[y_\mathcal{C} = u^a \;\middle|\; \begin{array}{l} \mathsf{par} = (N, \varphi(N)) \leftarrow \mathsf{GrpGen}(1^\lambda), \\ u \xleftarrow{\$} \mathbb{Z}_N, a \xleftarrow{\$} \mathbb{Z}_{\varphi(N)}, \\ (y_\mathcal{C}, y_\mathcal{S}, tr) \leftarrow (\mathcal{C}(1^\lambda, \mathsf{par}, a, u), \mathcal{S}(1^\lambda)) \end{array} \right] = 1.$$

We define two (weak) security notions that we call *instance-hiding* and *exponent-hiding* and which are variants of the similar notion used in [13, 14]. They formalize the idea that a passive adversary should not be able to retrieve the base and the exponent or the exponent (respectively) used in one or several runs of the delegation protocol. Exponent-hiding is naturally weaker than instance-hiding. Both notions are weaker than the indistinguishability notion from [9, 13, 14] which is itself weaker than the simulation-based security notion from [3]. The protocols from [2] and [1] were claimed to achieve the latter notion but we will actually show that they do not even achieve our weak instance-hiding and exponent hiding security properties (respectively).

The probabilistic computational experiments described in Figure 1 provide a formal description of these security notions. For the instance-hiding no-

Experiment $\mathbf{Exp}_{ih}(\mathcal{A},\mathcal{C},\lambda)$

$\mathsf{par} = (N,\varphi(N)) \leftarrow \mathsf{GrpGen}(1^\lambda)$
$(a,u) \xleftarrow{\$} \mathbb{Z}_{\varphi(N)} \times \mathbb{Z}_N$
$i \leftarrow 0$, $tr_0 \leftarrow \emptyset$
$(\alpha, \mathsf{ch\_base}, \mathsf{ch\_exp}, aux, (a_0,u_0)) \leftarrow \mathcal{A}(1^\lambda, \mathsf{par})$
while $\alpha \neq \mathsf{stop}$ do
  if $\mathsf{ch\_base} = \mathtt{true}$ then
    $u_i \leftarrow u$
  if $\mathsf{ch\_exp} = \mathtt{true}$ then
    $a_i \leftarrow a$
  $(y_i, (\alpha, \mathsf{ch\_base}, \mathsf{ch\_exp}, aux, (a_{i+1},u_{i+1})), tr_i)$
    $\leftarrow (\mathcal{C}(1^\lambda, \mathsf{par}, (a_i,u_i)), \mathcal{A}(aux))$
  $i \leftarrow i+1$
Return 1 if $(a_i,u_i) = (a,u)$ and 0 otherwise

Experiment $\mathbf{Exp}_{eh}(\mathcal{A},\mathcal{C},\lambda)$

$\mathsf{par} = (N,\varphi(N)) \leftarrow \mathsf{GrpGen}(1^\lambda)$
$a \xleftarrow{\$} \mathbb{Z}_{\varphi(N)}$
$i \leftarrow 0$, $tr_0 \leftarrow \emptyset$
$(\alpha, \mathsf{ch\_exp}, aux, (a_0,u_0)) \leftarrow \mathcal{A}(1^\lambda, \mathsf{par})$
while $\alpha \neq \mathsf{stop}$ do
  if $\mathsf{ch\_exp} = \mathtt{true}$ then
    $a_i \leftarrow a$
  $(y_i, (\alpha, \mathsf{ch\_exp}, aux, (a_{i+1},u_{i+1})), tr_i)$
    $\leftarrow (\mathcal{C}(1^\lambda, \mathsf{par}, (a_i,u_i)), \mathcal{A}(aux))$
  $i \leftarrow i+1$
Return 1 if $a_i = a$ and 0 otherwise

FIGURE 1. Random Experiment for Instance-Hiding and Exponent-Hiding Security Notions

tion, given a residue group description $(N,\varphi(N))$ output by $\mathsf{GrpGen}(1^\lambda)$ for some security parameter λ, a base $u \in \mathbb{Z}_N$ and an exponent $a \in \mathbb{Z}_{\varphi(N)}$ are picked uniformly at random. The adversary $\mathcal{A}$ is then allowed to run the delegation protocol multiple times with arbitrary inputs of its choice (a_i,u_i) in such a way that it can set $a_i = a$ (if $\mathsf{ch\_exp} = \mathtt{true}$) or $u_i = u$ (if $\mathsf{ch\_base} = \mathtt{true}$). Setting both flags to true, the delegation protocol is run on the input challenge (a,u) and setting both flags too false, the delegation protocol is run on some instance completely chosen by $\mathcal{A}$ (but it may obtain some information from such executions). The memory of the adversary for these multiple executions is stored in the variable aux. Eventually, the adversary $\mathcal{A}$ succeeds in this experiment if, when it decides to stop by setting the flag α to stop, it outputs the challenge pair (a,u). The exponent-hiding security notion is defined similarly but focuses only on the exponent.

DEFINITION 2.2 (Instance-hiding and Exponent-hiding). *Let* GrpGen *be a group generator and let* $(\mathcal{C},\mathcal{S})$ *be a client-server protocol for a delegated computation of exponentiation for* GrpGen. *Let* $\tau : \mathbb{N} \to \mathbb{N}$ *and* $\varepsilon : \mathbb{N} \to [0,1]$ *be two functions. We say that* $(\mathcal{C},\mathcal{S})$ *satisfies* (τ,ε)-instance-hiding *(*resp. (τ,ε)-exponent-hiding*) if, for any algorithm* $\mathcal{A}$*, it holds that for all integer* $\lambda \in \mathbb{N}$

$$\Pr[\nu = 1 | \nu \leftarrow \mathbf{Exp}_{ih}(\mathcal{A},\mathcal{C},\lambda)] \leq \varepsilon(\lambda)$$
$$(\text{resp.} \quad Pr[\nu = 1 | \nu \leftarrow \mathbf{Exp}_{eh}(\mathcal{A},\mathcal{C},\lambda)] \leq \varepsilon(\lambda) \quad)$$

where $\mathbf{Exp}_{ih}(\mathcal{A},\mathcal{C},\lambda)$ *(*resp $\mathbf{Exp}_{eh}(\mathcal{A},\mathcal{C},\lambda)$*). is the computational random experiment described in Figure 1 in which* $\mathcal{A}$ *runs in time at most* $\tau(\lambda)$.

The interactive protocol $(\mathcal{C},\mathcal{S})$ *is deemed instance-hiding-secure (*resp. *exponent-hiding-secure) if for all polynomial* $\tau : \mathbb{N} \to \mathbb{N}$ *and all* $\varepsilon : \mathbb{N} \to [0,1]$*, if* $(\mathcal{C},\mathcal{S})$ *does not satisfy* (τ,ε)*-instance-hiding (*resp. (τ,ε)*-exponent-hiding), then* ε *is negligible.*

Another important security definition called *verifiability* requires that the client should not accept a wrong value for the exponentiation which is delegated to the server. A delegation protocol that does not ensure verifiability may cause severe security problems (in particular if the exponentiation computation occurs in the verification algorithm of some authentication protocol).

DEFINITION 2.3 (Verifiability). *Let* GrpGen *be a group generator and let* $(\mathcal{C},\mathcal{S})$ *be a client-server protocol for a delegated computation of exponentiation for* GrpGen. *Let* $\tau : \mathbb{N} \to \mathbb{N}$ *and* $\zeta : \mathbb{N} \to [0,1]$ *be two functions. We say that* $(\mathcal{C},\mathcal{S})$ *satisfies* (τ,ζ)-verifiability *if, for any algorithm* $\mathcal{A}$ *, it holds that for all integer* $\lambda \in \mathbb{N}$*,*

$$\Pr\left[y_{\mathcal{C}} = u^a \;\middle|\; \begin{array}{l} \mathsf{par} = (N,\varphi(N)) \leftarrow \mathsf{GrpGen}(1^\lambda), \\ u \xleftarrow{\$} \mathbb{Z}_N, a \xleftarrow{\$} \mathbb{Z}_{\varphi(N)}, \\ (y_{\mathcal{C}}, y_{\mathcal{S}}, tr) \\ \qquad \leftarrow (\mathcal{C}(1^\lambda, \mathsf{par}, a, u), \mathcal{A}(1^\lambda)) \end{array} \right] \geq \zeta(\lambda).$$

where $\mathcal{A}$ *runs in time at most* $\tau(\lambda)$.

In practice, it is actually important to achieve a stronger notion of verifiability in which the adversary is allowed to run several executions of the delegation protocol with the client (for the same parameters $\mathsf{par} = (N,\varphi(N))$) before trying to make them accept a wrong value in a subsequent execution. We say that $(\mathcal{C},\mathcal{S})$ satisfies (τ,ζ)-*strong verifiability* if when $\mathcal{A}$'s total running time is upper-bounded by $\tau(\lambda)$ (over all their executions), then the probability that $\mathcal{C}$ outputs an erroneous value for u^a is at least $\zeta(\lambda)$.

2.3. Exponentiation Delegation: Protocols

In this section, $\alpha,\beta,\gamma \geq 0$ denote real parameters specifying the sizes of various variables in the delegation protocols.

We first provide a short description of the classical "textbook" RSA public-key encryption scheme [31]:

Key Generation: On input a parameter $\lambda \in \mathbb{N}$, the algorithm picks uniformly at random two distinct prime numbers p and q of bit-length λ. It then computes $N = pq$ and $\varphi(N) = (p-1)(q-1)$ the Euler's totient function of N. It picks an integer $e \in \{1, \ldots, \varphi(N)\}$ coprime with $\varphi(N)$ and computes the integer $d \in \{1, \ldots, \varphi(N)\}$ such that $ed \equiv 1 \bmod \varphi(N)$.

The key-generation algorithm outputs (N, e) as the public-key and (N, d) as the private key.

Encryption: To encrypt a plaintext $m \in \mathbb{Z}_N$ for a public key (N, e), the encryption algorithm outputs $c = m^e \bmod N$.

Decryption: To decrypt a ciphertext $c \in \mathbb{Z}_N$ with a private key (N, d), the decryption algorithm outputs $m = c^d \bmod N$.

The computation of $m = c^d \bmod N$ in the decryption algorithm requires $\mathcal{O}(\log d) = \mathcal{O}(\log N)$ multiplications in $\mathbb{Z}_N$. In some settings, this is too costly for a weak client and so the client can delegate part of this computation to a more powerful server. Obviously, the client wants to do so without revealing its private key d to the server.

2.3.1. RSA Delegation with Additive Splitting

In the *server-aided protocol with additive splitting* (see Figure 2), the client picks uniformly at randomly k in $\{1, \ldots, N^\alpha\}$ and s in $\{1, \ldots, N^\beta\}$, and computes the integer

$$A = d - s + k \cdot \varphi(N).$$

The client then sends c, A and N to the server which computes $m' = c^A \bmod N$ and sends it back to the client. The client eventually computes and outputs $m = m' \cdot c^s \bmod N$ as the decryption of c.

The number of modular multiplications to be performed by the client is proportional to the bit-length of s (and thus the smaller is β, the more efficient is the delegation protocol). In this protocol, the base m is sent in clear to the server, so the instance-hiding and exponent-hiding security notions are identical. Since the value returned by the server is not checked in any way, the protocol does not achieve verifiability. Mefenza and Vergnaud [15] pointed out that the best known attack on RSA with *partial key exposure* due to Joye and Lepoint [32] can be used to attack the protocol with a single run of the delegation protocol (see below).

2.3.2. The Su-Zhang-Xue (MCExp) Protocol

Recently, Su, Zhang and Xue [1] presented a protocol called MCExp for privately and verifiably outsourcing a modular exponentiation $u^a \bmod N$ where N is an RSA modulus, a is some private exponent (but not necessarily the inverse of a public value modulo $\varphi(N)$ as in the previous section) and u is some private base.

In the MCExp protocol, the client picks two random secret values r, s in $\{2, \ldots, 11\}$, and two random integers k_1 and k_2 of unspecified size. We assume that they are picked uniformly at random in $\{0, \ldots, N^\alpha\}$ and Figure 3 presents a generalized version of the MCExp protocol where s is picked uniformly at random in $\{0, \ldots, N^\beta\}$ and r is picked uniformly at random in $\{0, \ldots, N^\gamma\}$ (for some parameters β and γ).

The client then computes $A_1 = a - s + k_1 \cdot \varphi(N)$ and $A_2 = a \cdot r + k_2 \cdot \varphi(N)$, and sends them (in a random order) together with ω_1, ω_2 (two bases related to u) and the modulus N to the server which computes $R_1 = w_1^{A_1} \bmod N$ and $R_2 = w_2^{A_2} \bmod N$ and two other[4] modular exponentiations R_3 and R_4 (with bases and exponents which are not relevant for our attacks). It then sends all of this back to the client, which eventually computes u^a (and uses the other values to check the validity of the computation). The values of A_1 and A_2 are thus known to the server, but not their order.

The values r and s are not known by the server and are used to check the validity of the output u^a. More precisely, the client checks whether an equality of the form

$$\left(R_3^{t_1} R_1 S_1 u^s\right)^r = R_4 R_2 S_2 \tag{1}$$

holds (where t_1 is a "small" (private) exponents known by the client and S_1, S_2 are pre-computed values in $\mathbb{Z}_N$) and if it is the case, they output

$$R_3^{t_1} R_1 S_1 u^s$$

as the purported value for u^a. Note that if, an adversary can guess the integer r and identifies correctly R_1, R_2, R_3 and R_4, they can then set $\tilde{R}_1 = R_1 \cdot v$ and $\tilde{R}_2 = R_2 \cdot v^r$ for an arbitrary element $v \in \mathbb{Z}_N$; the 4-tuple $(\tilde{R}_1, \tilde{R}_2, R_3, R_4)$ then still satisfies (1) and the client will output

$$R_3^{t_1} \tilde{R}_1 S_1 u^s = R_3^{t_1} R_1 v S_1 u^s = u^a \cdot v$$

which is different from u^a for $v \neq 1$. Su *et al.* [1, *Theorem 1.8*] claim that their scheme achieves (τ, α)-verifiability for an unbounded τ and a constant α equal to 119/120. The previous remark shows that if a malicious server correctly guesses the integer r and the values R_1 and R_2, then they can make the client accept an incorrect value. For MCExp protocol with $r \in \{2, \ldots, 11\}$, this occurs with probability $1/10 \cdot 1/4 \cdot 1/3 = 1/120$; as such [1, *Theorem 1.8*] therefore claims that this is the best possible attack.

In the following, we present several attacks against the privacy of the MCExp protocol (and variants with larger parameters). Namely, we show that it does not

[4] In [1], it is mentioned that the four exponents are sent in a random order. In this paper, we assume that we can distinguish A_1 and A_2 from the two other exponents (which is natural since they have *a priori* different bit-lengths). If the four exponents are of the same bit-length this increase only the complexity of the attacks from Section 4 by a constant factor 6.

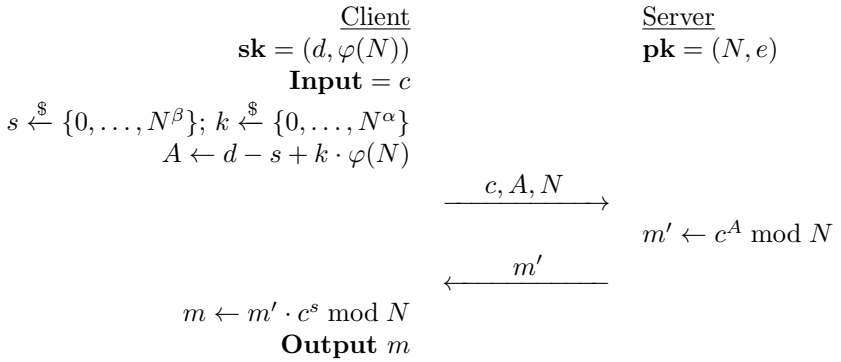

FIGURE 2. Server-aided RSA decryption with additive splitting

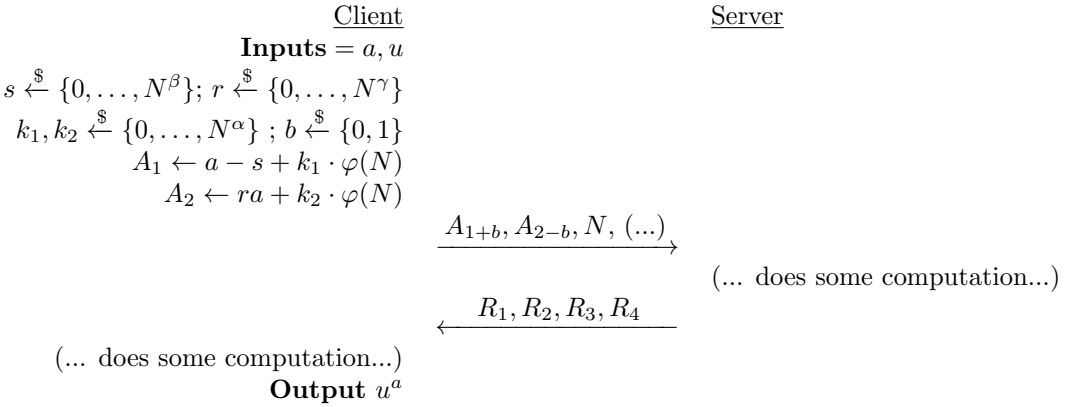

FIGURE 3. Partial description of the MCExp Protocol. The (...) notation means "some data" (not relevant in this paper).

achieve the exponent-hiding security notion (and thus that *Theorem 1.6* from [1] is also flawed). As a by-product of our attacks, an adversary is able to retrieve the Euler totient function $\varphi(N)$, r and s in addition to the exponent a. Therefore, we also show that the scheme proposed by Su *et al.* does not achieve (τ, α)-verifiability for $\tau = \text{poly}(n)$ and α larger than $11/12$.

2.3.3. The Rangasamy-Kuppusamy (MExpSOS) protocol

In this section, we briefly recall the protocol MExpSOS proposed by Rangasamy and Kuppusamy in [2]. In their protocol (see Figure 4), a client wants to privately and verifiably outsource a modular exponentiation $u^a \bmod N$ where N is either a prime number or an RSA modulus. Rangasamy and Kuppusamy made the very strong assumption that the modulus N is unknown to the adversary (whereas following Kerckhoff's principles [30], it is more natural to assume that the adversary knows the modulus since it is usually a part of the public parameters or the public key). Moreover, they add the privacy requirement that the delegation protocol should not reveal any information on N to a malicious server.

To mask the base u, the client generates an integer $L = \ell N$ which is a multiple of N (where ℓ is prime number large enough so that factoring L is assumed intractable) and sets $U = u + N' \mod L$ (where N' is another multiple of N) as the base for the delegated exponentiation. Obviously, this integer L has to be the same for all delegated computation since otherwise a simple greatest common divisor computation would reveal N as $\gcd(L, L') = \gcd(\ell N, \ell' N) = N$ for $\ell \neq \ell'$ two distinct prime numbers.

The client chooses two random integers r, s uniformly at random in $\{0, \ldots, N^\beta\}$ for some parameter $0 \leq \beta < 1$ as well as two integers k_1 and k_2 of unspecified size; we again assume that they are picked uniformly at random in $\{0, \ldots, N^\alpha\}$. It computes the values $A_1 = a - s + k_1\varphi(N)$ and $A_2 = a \cdot r + k_2\varphi(N)$ and sends them to the server in a random order, along with U and L. The server computes $R_1 = U^{A_1} \bmod L$ and $R_2 = U^{A_2} \bmod L$ and sends them back to the client. The client checks whether $(R_1 u^s)^r \equiv R_2 \bmod N$ and then recovers u^a as $u^a \equiv R_1 u^s \bmod N$.

As above, if an adversary can guess the integer r (and the correct order for R_1 and R_2), they can then set $\tilde{R_1} = R_1 \cdot v \bmod L$ and $\tilde{R_2} = R_2 \cdot v^r \bmod L$ for an arbitrary value x; the pair $(\tilde{R_1}, \tilde{R_2})$ passes the verification equation the client will output $u^a \cdot v \bmod N$ instead of $u^a \bmod N$. Rangasamy and Kuppusamy [2, *Theorem 2*] claim that their scheme achieves (τ, α)-

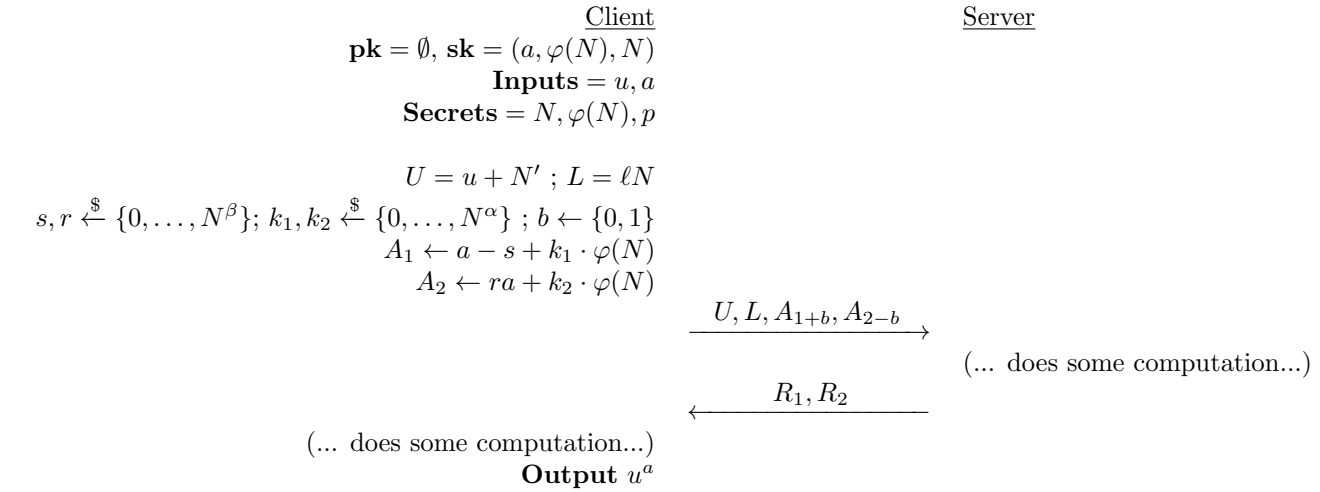

FIGURE 4. Partial description of the MExpSOS Protocol.

verifiability for an unbounded τ and α equal to $2N^\alpha$. The previous remark shows that if a malicious server correctly guesses the integer r and identifies R_1 and R_2, then they can make the client accept an incorrect value. This occurs with probability $2N^\alpha$ and [2, *Theorem 2*] thus claims that this attack is the best possible.

In the following, we present several attacks against the privacy of the MExpSOS protocol. Namely, we show that it does not achieve the instance-hiding security notion (and thus that *Theorem 1* from [2] is also flawed). As a by-product of our attacks, an adversary is able to retrieve the modulus N, the Euler totient function $\varphi(N)$, r and s in addition to the pair (a, u). Therefore, we also show that the scheme proposed by Rangasamy and Kuppusamy for useful parameters does not achieve (τ, α)-verifiability for $\tau = \text{poly}(n)$ and a non-trivial α.

3. LATTICE-BASED CRYPTANALYSIS

A lattice $\mathcal{L}$ is a discrete subgroup of $\mathbb{R}^n$ isomorphic to $\mathbb{Z}^\omega$ for some $\omega \in \mathbb{N}$. We denote by $\lambda_1(\mathcal{L}), \lambda_2(\mathcal{L})$, etc. the successive minima of the lattice and we define the volume of $\mathcal{L}$ by $\text{vol}(\mathcal{L}) = \sqrt{\det BB^t}$, where the rows of B span the lattice.

The LLL algorithm, due to Lenstra, Lenstra and Lovász [33] is one of the main tool of lattice-based cryptanalysis. We remind here some properties of a LLL-reduced basis, as seen in [34, *Chapter 2*] and refer the reader to [34] for additional details on lattice-based cryptanalysis.

THEOREM 3.1. *Let $(b_1, \ldots, b_\omega)$ be an LLL-reduced basis of a lattice $\mathcal{L}$ in $\mathbb{R}^n$ (with $\delta = 3/4$). Then*

1. $\|b_1\| \leq 2^{(\omega-1)/4}(\text{vol}\,\mathcal{L})^{1/\omega}$
2. *For all $i \in \{1, \ldots, \omega\}$, $\|b_i\| \leq 2^{(\omega-1)/2}\lambda_i(\mathcal{L})$*
3. $\|b_1\| \times \cdots \times \|b_\omega\| \leq 2^{\omega(\omega-1)/4}\,\text{vol}(\mathcal{L})$

From Theorem 3.1, we obtain a useful corollary:

LEMMA 3.1. *Let $\mathcal{L}$ be a lattice of dimension ω. In polynomial time, the LLL algorithm outputs two reduced basis vectors b_1 and b_2, that satisfy*

$$\|b_1\| \leq \|b_2\| \leq 2^{\frac{\omega}{4}}\,\text{vol}(\mathcal{L})^{\frac{1}{\omega-1}}$$

Proof. We can assume without loss of generality that $\|b_i\| \leq \|b_{i+1}\|$ in an LLL-reduced basis (we may swap basis vectors that violate this condition). Hence $\|b_1\| \times (\|b_2\|)^{\omega-1} \leq 2^{\omega(\omega-1)/4}\,\text{vol}(\mathcal{L})$.

Raising to the power $1/(\omega-1)$, we obtain: $(\|b_1\|)^{1/(\omega-1)} \times \|b_2\| \leq 2^{\omega/4}\,\text{vol}(\mathcal{L})^{1/(\omega-1)}$.

All we have to do left is noticing that $\|b_1\|$ is greater than 1 (we assume that the input basis has integer coordinates). □

The *Gaussian heuristic* "predicts" that if $\mathcal{L}$ is a full-rank lattice and C is a measurable subset of $\mathbb{R}^n$, then the number of points of $\mathcal{L} \cap C$ is roughly $\text{vol}(C)/\,\text{vol}(\mathcal{L})$. In particular, this asserts that the size of the shortest (non-zero) vector of $\mathcal{L}$ should be close to $\sqrt{n}\,\text{vol}(\mathcal{L})^{1/n}$.

3.1. Approximate Common Divisor Problem

In this section, we recall a simple approach for solving the so-called *approximate common divisor* problem. It relies on a simple lattice-based algorithm due to Lagarias [26] for simultaneous Diophantine approximation. The technique was first proposed by Howgrave-Graham in [27], then expanded in [28].

We consider an adversary that is given t integers $x_i = pq_i + r_i$ where p is an n-bit integer, q_i is an αn-bit integer and r_i is a βn-bit integer, where $\beta < 1$. Since the r_i's are "small", the x_i are approximate multiples of p and we have $x_i/x_0 \simeq q_i/q_0$ for $i \in \{1, \ldots, t-1\}$. In other words, the rationals q_i/q_0 are simultaneous rational approximations with the same denominator of the $t-1$ rationals x_i/x_0 for $i \in \{1, \ldots, t-1\}$. If the adversary can compute one such approximation, it can obtain q_0 and then r_0 as $x_0 \bmod q_0$ and p as $(x_0 - r_0)/q_0$.

This means that we are looking for $q_0, \ldots, q_t$ such that $x_i/x_0 \simeq q_i/q_0$ for all $i \in \{1, \ldots, t-1\}$. This can be rewritten by saying that $q_0 x_i - q_i x_0 = q_0 r_i - q_i r_0$ must be "small". Following [26], the adversary can thus construct the integer matrix:

$$M = \begin{pmatrix} 2^{\beta n+1} & x_1 & x_2 & \ldots & x_{t-1} \\ 0 & -x_0 & 0 & \ldots & 0 \\ 0 & 0 & -x_0 & \ldots & 0 \\ & & & \ddots & \\ 0 & 0 & 0 & \ldots & -x_0 \end{pmatrix} \quad (2)$$

The rows of M generate a lattice $\mathcal{L}$ of rank t that contains the "short" vector $v = (q_0, q_1, \ldots, q_{t-1})M = (q_0 2^{\beta n+1}, q_0 r_1 - q_1 r_0, \ldots, q_0 r_{t-1} - q_{t-1} r_0)$.

Each coordinate of v is less than $2 \times 2^{n(\alpha+\beta)}$ in absolute value, so the norm of v is upper-bounded by $2\sqrt{t} \times 2^{n(\alpha+\beta)}$. The volume of the lattice is $\mathrm{vol}(\mathcal{L}) = |\det(M)| = 2^{\beta n+1} x_0^{t-1} \approx 2^{\beta n+(t-1)(\alpha+1)n}$. Therefore if

$$2\sqrt{t} \times 2^{n(\alpha+\beta)} < \sqrt{t} |\det(M)|^{1/t}$$

then we expect by the Gaussian heuristic that v is a shortest non-zero vector in $\mathcal{L}$ (i.e. $\lambda_1(\mathcal{L}) = \|v\|$) and that the second minima $\lambda_2(\mathcal{L})$ should be close to

$$\sqrt{t} |\det(M)|^{1/t} \approx \sqrt{t} 2^{(\beta n+(t-1)(\alpha+1)n)/t}.$$

Running the LLL algorithm on the matrix M yields a reduced basis whose first vector is guaranteed to be at most $2^{t/2}$ times longer than the norm of v the (heuristically) shortest vector of the lattice (Theorem 3.1, item 2). Therefore, if

$$2^{t/2}\|v\| \leq 2\sqrt{t} \times 2^{t/2} 2^{n(\alpha+\beta)} < \sqrt{t} 2^{\beta n/t+(t-1)(\alpha+1)n/t}$$

the length of the first vector output by LLL is strictly less than $\lambda_2(\mathcal{L})$ and it follows that the LLL algorithm will (heuristically) return a shortest vector of $\mathcal{L}$.

If we forget about the constant terms, then this yields a condition on the exponents:

$$t^2 - 2nt(1-\beta) + 2n(1-\beta+\alpha) < 0$$

The discriminant $\Delta = 4n^2((1-\beta)^2 - 2(1-\beta+\alpha)/n)$ is necessarily positive for large enough n, so the inequation holds between the two roots:

$$t_\pm = n(1-\beta)\left(1 \pm \sqrt{1 - 2\frac{1-\beta+\alpha}{n(1-\beta)^2}}\right).$$

We focus on the smallest possible value of t satisfying the constraints. Using the Taylor expansion $1 - \sqrt{1+2x} = x + \mathcal{O}(x^2)$ near zero, we find that the small root is:

$$t_- = 1 + \frac{\alpha}{1-\beta} + \mathcal{O}(1/n).$$

This is the minimal required value of t so that the attack works using the LLL algorithm, and thus in polynomial time — subject to the condition that $\Delta > 0$, or equivalently $n > 2(1-\beta+\alpha)/(1-\beta)^2$.

3.2. Coppersmith's Methods

Some of our attacks requires the ability to find "small roots" of multivariate polynomials over the integers. To retrieve these values, we use variants of Coppersmith's methods [16, 17] presented in [25], that we now briefly present.

Let us consider a polynomial in $\mathbb{Z}[x, y, z]$:

$$f(x, y, z) = \sum_{i=1}^{\omega} a_i x^{d_i} y^{e_i} z^{f_i}.$$

We assume that there is "small root" (x_0, y_0, z_0) such that $f(x_0, y_0, z_0) = 0$ and that we are given bounds X, Y and Z such that $|x_0| < X, |y_0| < Y$ and $|z_0| < Z$.

For $h(x, y, z) \in \mathbb{Z}[x, y, z]$, we denote $\|h(x, y, z)\|$ the Euclidean norm of the vector formed by its coefficients (i.e. the root square sum of the coefficients of monomials in the polynomial).

Lemma 3.2 (Howgrave-Graham). *Let $h(x, y, z) \in \mathbb{Z}[x, y, z]$ be the sum of at most ω monomials and let $X, Y, Z > 0$. For any $(x_0, y_0, z_0) \in \mathbb{Z}^3$ that satisfies $|x_0| < X, |y_0| < Y, |z_0| < Z$, if*

$$f(x_0, y_0, z_0) \equiv 0 \bmod c \text{ and } \|f(xX, yY, zZ)\| < \frac{c}{\sqrt{\omega}},$$

then $f(x_0, y_0, z_0) = 0$.

A possible strategy to find the small root (x_0, y_0, z_0) of a polynomial h consists in constructing a lattice $\mathcal{L}$ of dimension ℓ containing only polynomials g_i such that $g_i(x_0, y_0, z_0) \equiv 0 \bmod c$. Then we reduce the lattice basis using LLL, we extract the two first polynomials g_1, g_2 from the reduced basis. If they satisfy the bound of Lemma 3.2, then these polynomial vanish on (x_0, y_0, z_0) over $\mathbb{Z}$ (not modulo c). Under the *heuristic* assumption that the polynomials f, g_1, g_2 define an algebraic variety of dimension 0, then we can find our small root by solving a polynomial system in 3 variables and 3 equations (by computing a Groebner basis for example).

From Lemma 3.1, we know that the LLL algorithm returns two vectors satisfying $\|b_1\| \leq \|b_2\| \leq 2^{\frac{\omega}{4}} \mathrm{vol}(\mathcal{L})^{\frac{1}{\omega-1}}$. Thus, the condition $\sqrt{\omega} 2^{\frac{\omega}{4}} \mathrm{vol}(\mathcal{L})^{\frac{1}{\omega-1}} < c$ implies that polynomials corresponding to the two shortest reduced basis vectors of an LLL-reduced basis satisfy Howgrave-Graham's bound. In practice, we ignore terms that do not depend on c, and check only if $\mathrm{vol}(\mathcal{L}) \leq c^{\omega-1}$.

The difficulty is to optimize the value of c and to choose the correct polynomials g_i to spawn the lattice. This problem has been addressed by Ernst *et al.* [25]. Using polynomials g_i of the form

$$x^i y^j z^k f(x, y, z) X^l Y^v Z^w \text{ and } c \cdot x^i y^j z^k,$$

Denoting W the value $W = \|f(xX, yY, zZ)\|$, they obtain the following results:

THEOREM 3.2 ([25]). *With the above notations, for every $\varepsilon > 0$, there exist a W_0 such that, for every $W \geq W_0$, the following holds: there is a PPT algorithm that recovers a small root (x_0, y_0, z_0) of $f(x, y, z)$ if there exists $\tau \geq 0$ in the two following cases:*

i) $f(x, y, z) = a_0 + a_1 x + a_2 y + a_3 yz$ *and*

$$X^{1+3\tau} Y^{2+3\tau} Z^{1+3\tau+3\tau^2} < W^{1+3\tau-\varepsilon}.$$

ii) $f(x, y, z) = a_0 + a_1 x + a_2 y + a_3 z + a_4 yz$ *and*

$$X^{2+3\tau} Y^{3+6\tau+3\tau^2} Z^{3+3\tau} < W^{2+3\tau-\varepsilon}.$$

4. ATTACKS AGAINST MCEXP USING A SINGLE DELEGATION

In this section, we describe polynomial-time attacks against the MCExp protocol for the exponent-hiding security notion after the adversary has passively observed a single exponentiation delegation. The attacks completely break the scheme since they recover in addition the (secret) factorization of N. The different attacks for the different variants of the protocol MCExp are summarized in Table 1.

4.1. A Straightforward Attack on the MCExp Protocol

As mentioned in the description of the protocol, the random masks s and r are picked in the very small set of integers $\{2, 3, \ldots, 11\}$. It is therefore possible to exhaustively search their values. The adversary observe A_{1+b} and A_{2-b}. They do not know b, but there are only two possibilities to correctly identify A_1 and A_2, so that any attack only has to be repeated twice. Once the adversary has guessed the values of b, r and s, they can then compute

$$r(A_1 - s) - A_2 = (rk_1 - k_2)\varphi(N),$$

and therefore learn a multiple of $\varphi(N)$. Note that there are only 200 possible values (10 for s, 10 for r, two possible choices for b) and that they can be explicitly computed.

In [29], Rabin provided a probabilistic polynomial-time algorithm which given an RSA modulus $N = pq$ and a multiple of its Euler's totient function $\varphi(N)$, outputs the factorization (p, q) in expected polynomial time $O(\log(N)^3)$. This in turns allows to reconstruct the secret key (by running the same algorithm as the legitimate user).

All that is left to do is to run Rabin's algorithm on the 200 candidates multiple of $\varphi(N)$: one of them is going to reveal the factorization of the RSA modulus N. Once the adversary knows the factorization of N and thus $\varphi(N)$, it also knows the values b, r and s with certainty and from A_1 it can simply retrieve the (secret) exponent a as $[A_1 \bmod \varphi(N)] + s$. The MCExp protocol therefore does not achieve (τ, ε)-exponent hiding for $\tau = \Omega(\log(N)^3)$ and $\varepsilon = 1$.

As explained in Section 2.3.2, when the adversary knows b and r, they can make the client accept an incorrect value as the result for the delegated exponentiation with probability 1. The MCExp protocol therefore does not achieve (τ, ζ)-verifiability for $\tau = \Omega(\log(N)^3)$ and $\zeta = 0$.

We implemented this attack for a modulus N of 2048 bits and $\alpha = 1$ (i.e., $k_1, k_2 \simeq N$): it runs in only 1.7s on a laptop.

4.2. Generalizations of MCExp

In its original presentation, the MCExp protocol is thus practically broken, mainly because of the bad design decision to limit r and s to extremely small numbers. In the rest of this paper, we show that the problem is in fact more fundamental.

As described in Figure 3, we now consider a generalization of the MCExp protocol where the two parameters r and s can be chosen in larger sets that prevent exhaustive search. More precisely, we assume that s and r are integers picked uniformly at random with $0 \leq s < N^\beta$ and $0 \leq r < N^\gamma$ where $\beta, \gamma \geq 0$ are constant parameters. Because the client must perform two exponentiations with exponents s and r, it follows that $\beta + \gamma$ should be quite smaller than 1, otherwise the computational load of the client is not significantly reduced and the protocol misses the point.

Let $t = p + q - 1$ so that $\varphi(N) = N - t$. When p and q are balanced, it always holds that $t \leq 3N^{1/2}$. Let us also denote by $k := rk_1 - k_2$, which is unknown.

When the attacker can only observe a single delegation, a simple attack strategy consists in solving the following non-linear system over the integers:

$$\begin{cases} rA_1 - A_2 - kN &= rs - kt \\ |s| &\leq N^\beta \\ |r| &\leq N^\gamma \\ |t| &\leq 3N^{1/2} \\ |k| &\leq N^{\alpha+\gamma} \end{cases} \quad (3)$$

The actual secret parameters (s, r, t, k) are a solution of this system. If this solution is unique, then solving (3) guarantees the recovery of all secrets. This in turn allows the adversary to factor N by computing $\varphi(N)$ and solving the quadratic equation $X^2 - tX + N = 0$ (whose roots are p and q).

However, depending on the values of α, β, γ, this system may have more than one solution. For instance, if (s, r, t, k) satisfies the equation, then $(s + k, r, t + r, k)$ also does, assuming that $\alpha < \beta$ and $\gamma < 1/2$. In the sequel, we focus on cases where (3) has a single solution.

4.3. Attack using Euclidean Division

Let us assume that r is *known*. Define $B := rA_1 - A_2$, whose value is also known. Equation (3) becomes:

§	Type	α	β	γ
4.1	Straightforward	arb.	$\simeq 0$	$\simeq 0$
4.3	Euclidean division	$\leq 1/2$	$< \alpha$	$\simeq 0$
4.4	Linearization	$\leq 1/2$	$< \alpha$	$< 1/4 - \alpha/2$
4.5	Coppersmith	$> 1/2$	$\lessapprox (\alpha + 1 - \sqrt{4\alpha^2 + 2\alpha - 2})/3$	$\simeq 0$
4.6	Coppersmith	arb.	$\simeq 0$	$\lessapprox (7 - 2\sqrt{7 + 6\alpha})/6$

TABLE 1. Summary of the attacks against the MCExp protocol

$B = rs + k\varphi(N)$. If rs is sufficiently small, then this implies that $B \simeq kN$, hence the quotient B/N should be close to k. Indeed, dividing by N yields:

$$B/N = (rs + kN - kt)/N = k + (rs - kt)/N$$

So the quotient of the Euclidean division of B by N is

$$\lfloor B/N \rfloor = k + \left\lfloor \frac{rs - kt}{N} \right\rfloor$$

If $0 \leq rs - kt < N$, then $\lfloor B/N \rfloor = k$. If $-N \geq rs - kt$, then $\lfloor B/N \rfloor = k - 1$. By assumption on the efficiency of the delegation protocol, we have $\beta + \gamma < 1$ and thus $rs \leq N^{\beta+\gamma} < N/2$ (for λ sufficiently large). If we assume moreover that $\alpha + \gamma < 1/2$, then we have $kt = (rk_1 - k_2)t \leq N^\gamma N^\alpha \cdot 3N^{1/2} < N/2$ (for λ sufficiently large) and we easily recover k as $\lfloor B/N \rfloor$ or $\lfloor B/N \rfloor + 1$..

Then, once k is known, we have $B - kN = rs - tk$. If $\beta + \gamma < \alpha$, we have $rs < k$ with overwhelming probability, and we can recover both rs and t by computing the Euclidean division of $B - kN$ by k. With t, we obtain $\varphi(N)$, and therefore we can factor N in polynomial time by solving a quadratic equation.

An interesting application is the following. Assume that r is small enough to be guessed by the adversary (which is the case when $2 \leq r \leq 11$ in the original MCExp protocol), and that $\beta < \alpha < 1/2$. The attack presented in this section shows that observing a single delegation is enough to factor N instantly in this setting in time $O(\log(N)^2)$. As in Section 4.1, the knowledge of r, p, q is sufficient for the adversary to compute s and then the secret exponent a (and also to break the verifiability property of the protocol).

To sum up, the modified MCExp protocol where r is picked uniformly at random in $\{2, \ldots, 11\}$ and s is picked uniformly at random in $\{0, \ldots, N^\beta\}$ with $\beta < \alpha < 1/2$ does not achieve (τ, ε)-exponent hiding nor (τ, ζ)-verifiability for $\tau = \Omega(\log(N)^2)$, $\varepsilon = 1 - \text{negl}(\lambda)$ and $\zeta = \text{negl}(\lambda)$. This attack was tested successfully in practice with a modulus N of 2048 bits, k_1 and k_2 of 992 bits, s of 960 bits and $r = 11$; it takes 0.1ms.

4.4. Attack using Linearization

In this section, we present an attack on the modified MCExp protocol when r is unknown (i.e. picked in a set too large to allow exhaustive search). In this case, we could try to partially solve (3) the easy way, by getting rid of the non-linear term $rs - kt$. Indeed, a solution of (3) is also a solution of the integer linear program:

$$\begin{cases} |rA_1 - kN - A_2| & \leq & N^{\alpha+\gamma+1/2} \\ |r| & \leq & N^\gamma \\ |k| & \leq & N^{\alpha+\gamma} \end{cases} \quad (4)$$

It is not immediately obvious whether or not (4) admits other solution than the "right" values of (r, k), but it does not matter because there is a very efficient way to find the right ones.

Solving (4) amounts to finding a vector close to the vector $(0, A_2)$ in the Euclidean lattice $\mathcal{L}$ spanned by the rows of:

$$G = \begin{pmatrix} xN^{2\alpha+2\gamma} & A_1 \\ 0 & N \end{pmatrix}.$$

where $x > 1$ is a constant to be determined later.

We claim that if $\alpha + 2\gamma < 1/2$, then $(r, -k)G$ is (heuristically) the lattice vector closest to $(0, A_2)$. Because solving the *Closest Vector Problem* in dimension two can be done efficiently, it follows that r and k can both be retrieved efficiently when this condition is met. Combined with the "Euclidean division attack" of the previous section, this shows that a single delegation is sufficient to factor N in polynomial time even when r is not necessarily extremely small, when $\beta < \alpha$ and $\alpha + 2\gamma < 1/2$.

We now prove this claim. The volume of the lattice is $\text{vol}(\mathcal{L}) = xN^{1+2\alpha+2\gamma}$. By the Gaussian heuristic, we estimate that the length of the shortest vector is $\approx \sqrt{x}N^{1/2+\alpha+\gamma}$. Let d denote the distance between $(r, -k)G$ and $(0, A_2)$. We have

$$d^2 = r^2x^2N^{4\alpha+4\gamma} + (rA_1 - kN)^2,$$

which can be upper-bounded by

$$d^2 \leq x^2N^{4\alpha+6\gamma} + N^{1+2\alpha+2\gamma}.$$

By hypothesis, $N^{2\alpha+4\gamma}/N$ quickly goes to zero when N is large. Therefore, for large enough N, we have $d^2 \leq 2N^{1+2\alpha+2\gamma}$, regardless of the value of x. In fact, $x = \log N$, or even $x = N^\varepsilon$ (for a small enough ε) would also work.

Thus, for a sufficiently large x and for N large enough, the distance between $(0, A_2)$ and $(r, -k)G$ should always be smaller than the length of the shortest vector of the lattice, so that $(r, -k)G$ should be the lattice vector closest to $(0, A_2)$.

In the range of parameters $\beta < \alpha$ and $\alpha + 2\gamma < 1/2$, as in the previous section, this shows that the modified MCExp protocol does not achieve (τ, ε)-exponent hiding nor (τ, ζ)-verifiability for $\tau = \text{poly}(\lambda)$, $\varepsilon = 1 - \text{negl}(\lambda)$ and $\zeta = \text{negl}(\lambda)$ (assuming the Gaussian heuristic for the family of Euclidean lattice $\mathcal{L}$).This attack was tested successfully in practice with a modulus N of 2048 bits, k_1 and k_2 of 496 bits, s of 480 bits and r of 256 bits ; it takes 30ms.

4.5. Attack when s is Somewhat Large using Coppersmith's Methods

The two previous attacks require $\alpha \leq \frac{1}{2}$. In this section, we lift this restriction, and focus on the case where r is still very small (smaller than 11 as in the original protocol) but s is not. In this case, it is still possible to do an exhaustive search on r and thus we assume it to be known. We also assume that $\alpha > 1/2$ and that $\beta < \alpha$.

We show that it is heuristically possible, for a range of values of the parameters α and β, to recover in polynomial-time the factorization of N by finding small roots of a polynomial in three variable using the technique exposed in Section 3.2.

In this setting, $\lfloor B/N \rfloor$ is no longer equal to k or $k-1$, but $k^* = \lfloor B/N \rfloor$ is still a good *approximation* of k. Hence, we can rewrite k as $k = k^* + k'$. The new parameters k^* and k' are respectively upper-bounded by $N^{1/2}$ and $N^{\alpha - 1/2}$. Equation (3) is equivalent to $f(s, t, k') = 0$, where f is the tri-variate polynomial:

$$\begin{aligned} f(x, y, z) &= N \cdot (k^* + y) - z \cdot (k^* + y) + r \cdot x - B, \\ &= N \cdot k^* - B + r \cdot x - k^* \cdot z + Ny - k^* \cdot yz. \end{aligned}$$

We want to find a small root of f, expecting it to be (s, k', t). To make Coppersmith's method work, we first need to compute the bounds: $X \approx N^{\beta}$, $Y \approx N^{\alpha - 1/2}$, $Z \approx N^{1/2}$, and finally, $W \approx N^{1/2 + \alpha}$. By Theorem 3.2, finding the small root in polynomial time is feasible for parameters α, β such that there exists $\tau \geq 0$ satisfying:

$$X^{2+3\tau} Y^{3+6\tau+3\tau^2} Z^{3+3\tau} < W^{2+3\tau-\varepsilon}. \qquad (5)$$

The optimal τ (the one which will gives us the largest ranges for α and β) is $\tau = -\frac{\alpha+\beta-1}{2\alpha-1}$. We are thus led to distinguish two cases:

- if $\alpha + \beta > 1$, then τ is negative and the optimal value of τ is zero. In this case, (5) becomes $\alpha(\varepsilon + 1) + 2\beta < 1 - \varepsilon/2$ and no pair (α, β) satisfies both conditions;
- if $\alpha + \beta \leq 1$, then τ is positive and substituting it into (5) yields

$$4\alpha^2\varepsilon + \alpha^2 + 2\alpha\beta - 3\beta^2 + 2\beta - \varepsilon - 1 < 0.$$

This is true when

$$3\beta < \alpha + 1 - \sqrt{4\alpha^2(1 + 3\varepsilon) + 2\alpha - (2 + 3\varepsilon)}.$$

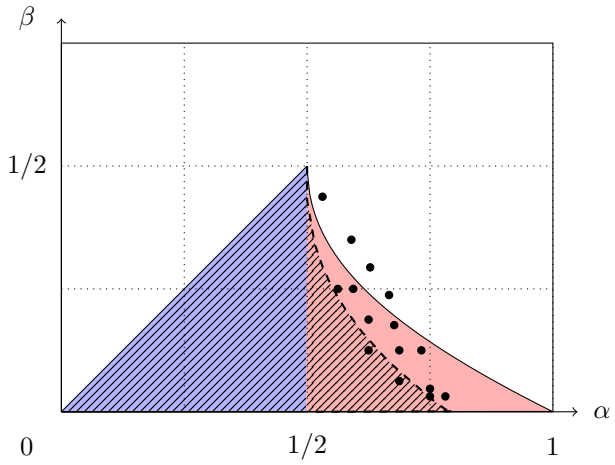

FIGURE 5. Attacks based on Coppersmith methods with small r. In blue the parameters that can be broken by the "Euclidean division" of Section 4.3, in red the parameters broken by the attack based on finding small roots of Section 4.5. The black dots mean that the attack has been carried out in practice when N is 2048-bit long. In the hatched area, the attack is known to be practically feasible for this particular size of N.

It must be noted that this bound is exactly the same bound as in the Joye-Lepoint attack against RSA with partial key-exposure [32] (see Section 4.7). For any given pair (α, β), there is a threshold W_0 such that the attack (heuristically) works for all $W > W_0$ (assuming that the generated polynomials define an algebraic variety of dimension 0). In practice, this means that it only works for "large enough" N. As in the previous sections, this is sufficient to (heuristically) break the exponent-hiding and verifiability properties of the modified MCExp protocol in polynomial-time for this range of parameters.

Experiments. Several factors make the practical assessment of the attack difficult. First of all, the above analysis implies that, given α and β, the attack only works for large enough N. It is not easy to determine what ranges of α, β are actually breakable for practical sizes of N.

We thus decided to fix the size of N to a practical value (2048 bits) and ran a series of experiments. The actual algorithm of [25] uses two integer parameters which are determined asymptotically in its analysis (summarized in Theorem 3.2). Running the algorithm in practice requires finding actual values of those parameters.It is fairly easy and fast to find automatically pairs such that finding the small root is (heuristically) guaranteed to succeed (for instance by exhaustive search over a small space). If N is too small, then no such pair may exist.

In Figure 5, the dashed line show the largest β for which parameters are known to exist and make the

attack work with this particular size of N. This is visibly well below the bound "for large enough N".

In practice, the algorithms performs better than what the theory predicts (it even succeeds for values of α, β outside of the promised area). This phenomenon is documented [35]. Smaller parameters may work, and manually tuning them could lead to better execution times or may allow the attack to work for a smaller N. This process is nevertheless error-prone and time-consuming.

We have tested the attack with k_1, k_2 of 1500 bits and s of 256 bits. It succeeded with $m = 4, t = 2$, and spent 64h LLL-reducing a lattice of dimension 133 with 14-Kbit coefficients.

4.6. Attack when r is Somewhat Large using Coppersmith's Methods

We finally consider the setting where one increases the range of the multiplicative mask r while s is kept small (less than 11).

We consider the polynomial:

$$f(x, y, z) = A_2 + x \cdot (A_1 - s) - yN - yz$$

It admits a small root (r, k, t), that we will find using the same method as before.

We first need to compute the bounds: $X \approx N^{\gamma}$, $Y \approx N^{\alpha+\gamma}$, $Z \approx N^{1/2}$ and finally, $W \approx N^{1+\alpha+\gamma}$. By Theorem 3.2, we are searching for $\tau \geq 0$ such that

$$X^{1+3\tau} Y^{2+3\tau} Z^{1+3\tau+3\tau^2} < W^{1+3\tau-\varepsilon}. \qquad (6)$$

Hence, we are searching for parameters $\alpha, \gamma, \tau \geq 0$ such that:

$$N^{\gamma(1+3\tau)} N^{(\alpha+\gamma)(2+3\tau)} N^{(1+3\tau+3\tau^2)/2} < N^{(1+\alpha+\gamma)(1+3\tau)}.$$

The optimal τ is $\tau = 1/2 - \gamma$, and with this value inequality (6) become

$$\alpha(1+\varepsilon) < \left(\frac{7}{8} - \varepsilon\right) - \left(\frac{7}{2} + \varepsilon\right)\gamma + \frac{3}{2}\gamma^2$$

Once again, once α and γ are given such that this constraint is satisfied for some ϵ, the attack will (heuristically) only work for large enough N. Again, this attack is sufficient to (heuristically) break the exponent-hiding and verifiability properties of the modified MCExp protocol in polynomial-time for this range of parameters.

Experiments. We tested the attack in practice (see Figure 6). It requires larger values of n to actually work than the previous one. When N is 2048-bit long, the attack worked with k_1, k_2 of 1152 bits and r of 64 bits, using $m = 4, t = 2$. It spent 3900s LLL-reducing a lattice of dimension 98 with 19-Kbit coefficients.

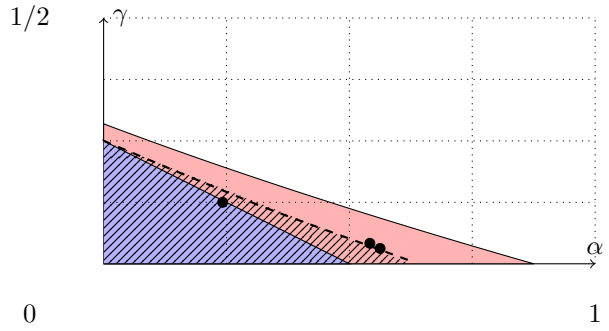

FIGURE 6. Attacks based on Coppersmith's methods. In blue, the parameters that can be attacked with the linearization attack of Section 4.4. In red, the parameters that can be attacked with the attack in this section. The black dots mean that the attack has been carried out in practice when N is 2048-bit long. In the hatched area, the attack is known to be practically feasible for this particular size of N.

4.7. Comparison With Known Attacks Against Server-Aided RSA Protocol with Additive Splitting

In the special case where the delegated exponentiation is an RSA signature (or an RSA decryption) with a secret exponent d, additional knowledge is available to the adversary, namely the fact that $ed \equiv 1 \bmod \varphi(N)$ for an exponent e which is usually part of the client's public key. In this case, the MCExp protocol can be seen as a special case of the server-aided RSA protocol with additive splitting (by just focusing on A_1 and ignoring A_2).

Both protocol reveal a "blinded" version of the secret key. Mefenza and Vergnaud [15] pointed out that the best known attack on RSA with *partial key exposure* due to Joye and Lepoint [32] can be used to attack the protocol, by extracting the secret exponent from A_1.

Theorem 4.1 (Joye-Lepoint attack). *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) from the observation of a single delegation, provided that:*

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

The size of the public exponent e intervenes in the complexity of the Joye-Lepoint attack, while it plays no role in our case. For instance, the Joye-Lepoint attack could be completely prevented by choosing a random e modulo $\varphi(N)$. The MCExp protocol thus introduces *additional* weaknesses compared to the simple server-aided protocol with additive splitting.

With a small public exponent, the complexity of the Joye-Lepoint attack is essentially the same as that of

the attack with small r of Section 4.5 — which is not surprising since it essentially solves the same problem. However, the Joye-Lepoint attack is not sensitive to the value of r, and thus could potentially break parameters that are out of the reach of all our attacks, such as $\alpha = 3/4, \beta = \gamma = 1/8$.

5. ATTACKS AGAINST MCEXP AND MEXPSOS WITH SEVERAL DELEGATIONS

In the previous section, we presented several attacks against the modified MCExp protocol using a single run of the protocol. They considerably reduce the range of parameters usable to obtain an efficient scheme secure in this limited setting. In this section, we present more devastating attacks in the case where the adversary is able to observe more than one exponentiation delegation (with independent random decomposition).

To prevent the attacks described in this section, a client can store the random decomposition used for each exponent or generate it as a pseudo-random function of the exponent. However, this approach increases the computational or memory complexity of the limited device and may not be usable in practice. Moreover, to counter *active attacks*, Pfitzmann and Waidner [36] suggested to renew the decomposition of the secret key. Renewing the random masks is also interesting to prevent side-channel attacks which are a major threat against implementations of cryptographic algorithms (especially on computationally limited devices).

The method is described for the server-aided RSA with additive splitting and we show that it applies to the MCExp and MExpSOS protocols.

5.1. Attack against MCExp with small r using Two Delegations

We first present another simple attack which applies when r is small enough to be guessed by the adversary and the client delegates two exponentiations for the same exponent a. This is for instance the case if the client performs two RSA signatures (even if the adversary does not know the associated public exponent).

In this setting, the adversary can recover the factorization of N in polynomial time, regardless of the size of s. As in the previous section, this approach breaks the exponent-hiding security notion of MCExp (with two runs in the random experiment $\mathbf{Exp}_{eh}(\mathcal{A}, \mathcal{C}, \lambda)$) for $\tau(\lambda) = \text{poly}(\lambda)$ and $\varepsilon(\lambda) = 1$ and its strong-verifiability (using the knowledge acquired after two sessions to obtain the factorization of N and the random mask r used in the second session, the adversary is able to make the client output an incorrect value).

We suppose that the adversary can obtain two delegations with the same a. We have

$$\begin{aligned} A_1 &= a - s + k_1\varphi(N) \\ A_2 &= ra + k_2\varphi(N) \\ A'_1 &= a - s' + k'_1\varphi(N) \\ A'_2 &= r'a + k'_2\varphi(N) \end{aligned}$$

We assume that we know r,r' and the order of A_1, A_2 and A'_1, A'_2 (which corresponds to an exhaustive research over only 400 possibilities) then we can compute $r'A_2 - rA'_2 = (r'k_2 - rk'_2)\varphi(N)$ and run Rabin's algorithm using this multiple of $\varphi(N)$ (as in Section 4.1).

5.2. Attack against Server-Aided RSA with Additive Splitting

We present a direct application of the approximate common divisor algorithm described in Section 3.1 to attack the server-aided protocol with additive splitting when the adversary is allowed to observe $t \geq 2$ different executions of the protocol.

The adversary observes a sequence of t transcripts $A_i \leftarrow d - s_i + k_i \cdot \varphi(N)$ for $i \in \{0, \ldots, t-1\}$, where each s_i and each k_i are chosen independently at random in their range. For all $i \in \{1, \ldots, t-1\}$, we simply compute $B_i = A_i - A_0$ (which cancels the "large" unknown d term), and we obtain

$$B_i = (q_i - q_0)\varphi(N) + (r_i - r_0)$$

for $i \in \{1, \ldots, t-1\}$. This is exactly an instance of the approximate common divisor problem, and the algorithm of Section 3.1 finds the approximate common divisor $\varphi(N)$ in polynomial time, provided that:

1. $\log(N) > 2(1 - \beta + \alpha)/(1 - \beta)^2$,
2. $t > 1 + \alpha/(1 - \beta)$.

With N of 2048 bits and s of 256 bits, the first condition translates to $\alpha < 783.125$. Observing $8\alpha/7 + \mathcal{O}(1)$ delegations are sufficient in practice to factor the modulus N. For instance, with $\alpha = 15$, so that A is 32Kbit-long, then observing 20 delegations is sufficient. The attack reduces a dimension-20 lattice with 22-Kbit coefficients, which takes about 15s.

Preventing the attack would require choosing α to be so large that the first condition is not satisfied. Using $\alpha = \log(N)/2$ is sufficient, but in this case the multiple k_i's have $\Omega(\log(N)^2)$ bits and the client must compute the product $k_i \cdot \varphi(n)$, which is more costly than performing the exponentiation directly.

5.3. Attack against MCExp with large s and r

The previous attack can readily be generalized to attack the modified MCExp protocol with large s and r. The technique consists in simply forgetting the A_2's and

using only the A_1's. The range of the r's is therefore irrelevant and the attack applies for all $\gamma > 0$. We are thus in the same setting as in the server-aided RSA protocol from Section 5.2, except that for each delegation, we have to guess the order of A_1 and A_2.

We have to observe $t = 1 + \alpha/(1-\beta)$ delegations, and try all the 2^t possible cases[5] and each trial takes polynomial time in $\log(N)$ and t. For $\alpha = O(1)$ and $\beta = O(1)$, this attack allows to obtain the factorization of N in polynomial time. As in Section 4.5, this approach breaks the exponent-hiding security notion of MCExp (with t runs in the random experiment $\mathbf{Exp}_{eh}(\mathcal{A}, \mathcal{C}, \lambda)$) for $\tau(\lambda) = \text{poly}(\lambda)$ and $\varepsilon(\lambda) = 1$ and its strong-verifiability (using the knowledge acquired after $t-1$ sessions to obtain the factorization of N and the random mask r used in the last session, the adversary is able to make the client output an incorrect value).

For more concrete parameters, with $\alpha = 1$ and both r, s of 256 bits, the attack requires observing 4 delegations. There are 16 possibles orders of the pairs (A_1, A_2), therefore the basic procedure has to be repeated 16 times. This takes 0.1s in total.

For the protocol to be useful, it is necessary that β be much smaller than 1. Making the attack difficult in practice requires a large value of α, for instance $\alpha = 128$. This adds to the computational load of the client. Indeed, delegating an RSA signature for a 2048-bit modulus then requires transmitting 0.5 Mbit to the server. It also requires the server to perform a large exponentiation (with an exponent of 262,144 bits, we measured a running time of 0.3s on a laptop for the exponentiation). Asymptotically, using $\beta = 1/2$ and $\alpha = \Omega(\log(\lambda)^2)$ for instance, makes our attack not polynomial-time (but it also makes the delegation scheme useless in practice).

5.4. Attack against MExpSOS

Finally, we show that the previous attack can also be applied to the protocol MExpSOS. The exponents sent by the client to the server MExpSOS are also of the form $A_1 = a - s + k_1\varphi(N)$ and $A_2 = a \cdot r + k_2\varphi(N)$ (in a random order) where a is the exponent of the delegated computation. As above, we have to observe $t = 1 + \alpha/(1-\beta)$ delegations, and try all the 2^t possible cases in order to obtain the value $\varphi(N)$. The difference is that the (adversarial) server does not know the modulus N itself but only some multiple $L = \ell N$.

If N is prime then $\varphi(N) = N - 1$ and the adversary can readily obtain N. If N is an RSA modulus, an adversary can also simply recovers N from $\varphi(N)$ and L. Indeed, for a random prime number ℓ (in the range that makes L difficult to factor) and a random integer $x \in \mathbb{Z}_N$, we have $x^{\varphi(N)} \equiv 1 \bmod N$ by Fermat's little theorem whereas $x^{\varphi(N)} \neq 1 \bmod \ell$ with overwhelming probability. Therefore, we have $\gcd(x^{\varphi(N)} - 1, L) = N$ with overwhelming probability. This is sufficient to obtain N and then factor it using the knowledge of $\varphi(N)$. In addition, when the adversary has obtained N, they can simply recover the base of the delegated exponentiation as $u = U \bmod N$. This approach therefore breaks the instance-hiding security notion of MExpSOS and its strong-verifiability.

We implemented this attack. For concrete parameters, with N a 2048-bit RSA modulus, ℓ a 1024-bit prime number (to prevent factoring attacks), $\alpha = 1$ and both r, s of 256 bits, the attack requires observing 4 delegations. There are 16 possibles orders of the pairs (A_1, A_2), therefore the basic procedure has to be repeated 16 times. This takes 0.3s in total. As above, to prevent this attack, using $\beta = 1/2$, the delegation protocol requires $\alpha = \Omega(\log(\lambda)^2)$ and this makes the delegation scheme useless in practice.

[5] If the four exponents used in MCExp are of the same bit-length (see Section 2.3.2), this increase the number of possible cases to 12^t but does not change the asymptotic complexity of our attack.

6. CONCLUSION

We presented several lattice-based attacks on two protocols recently proposed by Su, Zhang and Xue [1] and Rangasamy and Kuppusamy [2]. In both papers, the authors extended their approach and proposed protocols for simultaneous modular exponentiations. These extensions rely on the same masking techniques and our cryptanalytic methods also apply to these generalized protocols.

Chevalier *et al.* [13, 14] proved lower bounds on the efficiency of generic private exponentiation outsourcing protocols in prime order groups. These bounds suggest that for a variable base private modular exponentiation delegation (which is the case of interest for RSA decryption/signature), it is probably difficult to improve the client efficiency by more than a constant factor. It remains open to provide more efficient provably secure protocols and complexity lower bounds for exponentiation protocols in groups of unknown composite order. Another interesting problem is to provide lower bound on the communication and computational complexities of verifiable exponentiation delegation.

ACKNOWLEDGMENT

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

DATA AVAILABILITY STATEMENT

There are no new data associated with this article.

REFERENCES

[1] Su, Q., Zhang, R., and Xue, R. (2020) Secure outsourcing algorithms for composite modular exponentiation based on single untrusted cloud. *Comput. J.*, **63**, 1271.

[2] Rangasamy, J. and Kuppusamy, L. (2018) Revisiting single-server algorithms for outsourcing modular exponentiation. In Chakraborty, D. and Iwata, T. (eds.), *INDOCRYPT 2018*, December, LNCS, **11356**, pp. 3–20. Springer, Heidelberg.

[3] 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.

[4] 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.

[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] 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.

[8] 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.

[9] Cavallo, B., Crescenzo, G. D., Kahrobaei, D., and Shpilrain, V. (2015) Efficient and secure delegation of group exponentiation to a single server. In Mangard, S. and Schaumont, P. (eds.), *Radio Frequency Identification. Security and Privacy Issues - 11th International Workshop, RFIDsec 2015, New York, NY, USA, June 23-24, 2015, Revised Selected Papers*, Lecture Notes in Computer Science, **9440**, pp. 156–173. Springer.

[10] Crescenzo, G. D., Khodjaeva, M., Kahrobaei, D., and Shpilrain, V. (2019) Secure delegation to a single malicious server: Exponentiation in RSA-type groups. *7th IEEE Conference on Communications and Network Security, CNS 2019, Washington, DC, USA, June 10-12, 2019*, pp. 1–9. IEEE.

[11] 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.

[12] 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.

[13] 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.

[14] Chevalier, C., Laguillaumie, F., and Vergnaud, D. (2021) Privately outsourcing exponentiation to a single server: Cryptanalysis and optimal constructions. *Algorithmica*, **83**, 72–115.

[15] Mefenza, T. and Vergnaud, D. (2019) Cryptanalysis of server-aided RSA protocols with private-key splitting. *Comput. J.*, **62**, 1194–1213.

[16] 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.

[17] 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.

[18] 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.

[19] Blömer, J. and May, A. (2005) A tool kit for finding small roots of bivariate polynomials over the integers. In Cramer, R. (ed.), *EUROCRYPT 2005*, May, LNCS, **3494**, pp. 251–267. Springer, Heidelberg.

[20] 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.

[21] Herrmann, M. and May, A. (2009) Attacking power generators using unravelled linearization: When do we output too much? In Matsui, M. (ed.), *ASIACRYPT 2009*, December, LNCS, **5912**, pp. 487–504. Springer, Heidelberg.

[22] Bauer, A., Vergnaud, D., and Zapalowicz, J.-C. (2012) Inferring sequences produced by nonlinear pseudorandom number generators using Coppersmith's methods. In Fischlin, M., Buchmann, J., and Manulis, M. (eds.), *PKC 2012*, May, LNCS, **7293**, pp. 609–626. Springer, Heidelberg.

[23] May, A. and Ritzenhofen, M. (2008) Solving systems of modular equations in one variable: How many RSA-encrypted messages does eve need to know? In Cramer, R. (ed.), *PKC 2008*, March, LNCS, **4939**, pp. 37–46. Springer, Heidelberg.

[24] May, A. and Ritzenhofen, M. (2009) Implicit factoring: On polynomial time factoring given only an implicit hint. In Jarecki, S. and Tsudik, G. (eds.), *PKC 2009*, March, LNCS, **5443**, pp. 1–14. Springer, Heidelberg.

[25] 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.

[26] Lagarias, J. C. (1982) The computational complexity of simultaneous diophantine approximation problems. *23rd FOCS*, November, pp. 32–39. IEEE Computer Society Press.

[27] Howgrave-Graham, N. (2001) Approximate integer common divisors. *Cryptography and Lattices, International Conference, CaLC 2001, Providence, RI, USA, March 29-30, 2001, Revised Papers*, pp. 51–66.

[28] van Dijk, M., Gentry, C., Halevi, S., and Vaikuntanathan, V. (2010) Fully homomorphic encryption over the integers. In Gilbert, H. (ed.), *EUROCRYPT 2010*, May / June, LNCS, **6110**, pp. 24–43. Springer, Heidelberg.

[29] Rabin, M. O. (1979) Digitalized signatures and public-key functions as intractable as factorization. Technical report. Massachusetts Institute of Technology, USA.

[30] Kerckhoffs, A. (1883) La cryptographie militaire. *Journal des sciences militaires*, **9**, 5–83.

[31] Rivest, R. L., Shamir, A., and Adleman, L. M. (1978) A method for obtaining digital signatures and public-key cryptosystems. *Communications of the Association for Computing Machinery*, **21**, 120–126.

[32] 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.

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

[34] Nguyen, P. Q. and Vallée, B. (2009) *The LLL Algorithm: Survey and Applications*, 1st edition. Springer Publishing Company, Incorporated.

[35] 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.), *Public Key Cryptography - PKC 2010, 13th International Conference on Practice and Theory in Public Key Cryptography, Paris, France, May 26-28, 2010. Proceedings*, Lecture Notes in Computer Science, **6056**, pp. 53–69. Springer.

[36] 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.