A Complete and Explicit Security Reduction Algorithm for RSA-based Cryptosystems

Kaoru Kurosawa[1], Katja Schmidt-Samoa[2], and Tsuyoshi Takagi[2]

[1] Ibaraki University,
4-12-1 Nakanarusawa, Hitachi, Ibaraki, 316-8511, Japan
kurosawa@cis.ibaraki.ac.jp
[2] Technische Universität Darmstadt, Fachbereich Informatik,
Alexanderstr.10, D-64283 Darmstadt, Germany
{samoa,takagi}@informatik.tu-darmstadt.de

Abstract. In this paper, we introduce a conceptually very simple and demonstrative algorithm for finding small solutions (x, y) of $ax + y = c \bmod N$, where $\gcd(a, N) = 1$. Our new algorithm is a variant of the Euclidian algorithm. Unlike former methods, it finds a small solution whenever such a solution exists. Further it runs in time $\mathcal{O}((\log N)^3)$, which is the same as the best known previous techniques, e.g. lattice-based solutions.
We then apply our algorithm to RSA-OAEP and RSA-Paillier to obtain better security proofs. We believe that there will be many future applications of this algorithm in cryptography.

Keywords: Provable security, Euclidean algorithm, Lattice reduction, RSA cryptosystem.

1 Introduction

Lattice reduction algorithms have been successfully applied to many cases of modern cryptography. Especially, this methods allow us to find a small solution (x, y) of the linear modular congruence

$$ax + y = c \bmod N, \tag{1}$$

where the integers a and N are coprime, i.e. $\gcd(a, N) = 1$. This technique was used to prove the security of RSA-OAEP and RSA-Paillier.

By using the above mentioned technique, Fujisaki et al. showed that RSA-OAEP is semantically secure against adaptive chosen ciphertext attacks (IND-CCA2) under the RSA assumption in the random oracle model [FOPS01] after important works of [BR95,Sho02]. In the random oracle model, the OAEP conversion is a technique to design a secure encryption scheme from any trapdoor one-way permutation [BR95]. We write f-OAEP if f is the underlying trapdoor function. Today's most famous cryptosystem, RSA-OAEP, is a result of this work.

In the standard model, on the other hand, it is known that RSA-Paillier encryption scheme is semantically secure against chosen plaintext attacks (IND-CPA). After the work of [ST02], Catalano et al. proved that the one-wayness of RSA-Paillier is equivalent to that of RSA [CGHGN01] by using the above technique with $c = 0$.

Now it is an important aim in cryptography to improve security reduction proofs, because the proposed size of the security parameters of a cryptosystem is directly influenced by the reduction costs.

In this paper, we introduce a conceptually much simpler and demonstrative algorithm for finding small solutions (x, y) of eq.(1). Our new algorithm is a variant of the Euclidian algorithm. Unlike the lattice-based method, it exploits that the sought-after small solution is non-negative. Further, it runs in time $O((\log N)^3)$, which is the same as the lattice-based method.

We then apply our algorithm to the security proof of RSA-OAEP to enhance the advantage of the reduction algorithm. The proof of RSA-OAEP is divided into two parts [FOPS01]. The first part was to prove the semantic security of the general OAEP conversion scheme under the so-called partial-domain one-wayness of the underlying trapdoor permutation. The second part was to exploit the homomorphic properties of RSA function in order to show the equivalence of partial-domain one-wayness and full-domain one-wayness in the RSA case.

However, the second part does not work for all values of a of eq.(1). More precisely, it works if the lattice $L_{a,N} = \{(u, v) \in \mathbb{Z}^2 | au = v \bmod N\}$ contains no non-zero vector of length at most 2^{k_0+2}, where k_0 is the maximal bit-length of the sought-after small solution. Since there are approximately $\pi 2^{2k_0+4} < 2^{2k_0+6}$ lattices containing a non-zero vector shorter than 2^{k_0+2}, the number of bad values for a is bounded above by 2^{2k_0+6}. Obviously, this result is not optimal, especially if the bound k_0 is close to half of the bit-length of N. One reason for the non-optimal performance of the lattice-based method is that it does not exploit all the information given about the sought-after solution. Namely, it takes no advantage of the fact that the solution is non-negative, not only small in absolute value.

For this problem, we are able to upper-bound the number of bad values for a by 2^{2k_0+1} instead of 2^{2k_0+6}.

Finally for RSA-Paillier, we use our new algorithm to construct an alternative reduction proof, extending the important work of Catalano et al. [CNS02]. Based on the analysis of our algorithm, we give the exact security analysis while Catalano et al. gave only asymptotic results.

But we want to point out that the major aim of this paper is not the advancement of the reduction proofs of RSA-OAEP and RSA-Paillier, respectively. Indeed, the achieved improvements are not dramatic ones. In fact, the main objective of this paper is the introduction of a new algorithm for solving two-variable linear congruence with small solutions. We believe that there will be many future applications of this algorithm in cryptography. To confirm this assumption, we revisit the security proofs of RSA-OAEP and RSA-Paillier as two applications.

(Related works:) Note that this task is not a new one in cryptography. In 1985, De Jonge and Chaum developed an attack against some kinds of RSA signature schemes [JC86], which was enlarged in 1997 by Girault and Misarsky [GM97], [Kat01]. These attacks utilize an affine variant of the Euclidian algorithm for solving two-variable linear modular equations with small solutions. But it has to be stressed, that this algorithm may fail, even if small solutions exist.

If $c = 0$, it is possible to find small solutions by means of continued fractions. Again, the Euclidian algorithm is used. But as before, this method is only heuristic, i.e. it does not succeed with all input.

Our algorithm, on the contrary, works for arbitrary inputs.

This paper is organized as follows: In Section 2 the security reduction algorithms of the RSA-OAEP and the RSA-Paillier cryptosystem are reviewed. In Section 3 we present our proposed algorithm for solving a two-variable modular equation with small solutions. In Section 4 the proposed algorithm is applied to the RSA-OAEP and the RSA-Paillier cryptosystem. In Section 5 we state a concluding remark.

2 Security Reduction Algorithms of RSA-OAEP and RSA-Paillier

In this section, we review the reduction proofs of the semantic security of RSA-OAEP and the one-wayness of RSA-Paillier. In both cases we are confronted with the problem of finding small solutions of modular congruences. We sketch the existing solutions which utilize lattice reduction methods.

2.1 RSA-OAEP

Let $f : \{0,1\}^k \mapsto \{0,1\}^k$ be a one-way trapdoor permutation. The random oracle reduction proof of f-OAEP states that if there is a CCA2-adversary against the semantic security of f-OAEP with a non-negligible advantage and running time t, then we are able construct an algorithm $\mathcal{A}$ with the following abilities: On the input $f(s_1, s_2)$, $\mathcal{A}$ computes in time polynomial in t and in the number of the adversary's queries to the different oracles (decryption and hash) a set S, such that the probability of s_1 being an element of S is non-negligible, too. In few words, the semantic security of f-OAEP in the random oracle model is reduced to the partial-domain one-wayness of f.

Now we consider the case $f = \text{RSA}$. We will sketch how the partial-domain one-wayness of RSA is reduced to its full-domain pendant. First, we introduce some notations. If x is a natural number, we write $[x]^l$ for the l most significant bits and $[x]_l$ for the l least significant bits of the binary representation of x, respectively. Let N be a $k-$bit RSA modulus and $k_0 < k/2$. Suppose there is an algorithm $\mathcal{A}$ that on the input $C = m^e \bmod N$ returns a set S of size q containing the integer $x := [m]^{k-k_0}$. We show how to solve the RSA problem (compute m from $C = m^e \bmod N$) using $\mathcal{A}$ as a subroutine. Pick any $a \in \mathbb{Z}_N^\times$

at random and run $\mathcal{A}$ on the inputs C and $C' := Ca^e \bmod N$. Because of the homomorphic properties of the RSA function we know that C' is the encryption of $ma \bmod N$. Hence the two output-sets produced by $\mathcal{A}$ contain the $k-k_0$ most significant bits of m and $ma \bmod N$, respectively. We define $u := [m]^{k-k_0}, r := [m]_{k_0}, v := [ma \bmod N]^{k-k_0}$ and $s := [ma \bmod N]_{k_0}$. Thus, $m = u \cdot 2^{k_0} + r$ and $ma \bmod N = v \cdot 2^{k_0} + s$ holds, leading to

$$\begin{aligned} v \cdot 2^{k_0} + s &= a \cdot (u \cdot 2^{k_0} + r) \bmod N \\ &\Rightarrow ar = s + c \bmod N,\ c = (v - ua) \cdot 2^{k_0} \bmod N. \end{aligned} \tag{2}$$

Thus for each of the q^2 possible combinations u, v taken from the output-sets of the two $\mathcal{A}$-runs, we get a linear modular congruence in the two unknowns r and s, where $0 \le r, s < 2^{k_0} < \sqrt{N}$. Note that therefore the reduction cost is quadratic in q (the value q arises in the random oracle part of the RSA-OAEP security proof, namely q equals the number of $\mathcal{A}_{SS}$'s queries to one of the hash oracles, where $\mathcal{A}_{SS}$ is an adversary against the semantic security of the OAEP conversion scheme). This is the main reason why the RSA-OAEP security proof is not meaningful for real-life parameters. Of course, an improvement of the congruence-solving-step will not affect this problem. Hence it is an important future task to find a reduction proof where only one $\mathcal{A}$-run is needed.

In the following, we call x, y a *small* solution of the congruence (2) iff $0 \le x, y < 2^{k_0}$ holds. We explain how Fujisaki et al. find a small solution using the Gaussian reduction algorithm. This algorithm can be viewed as a generalization of the Euclidian algorithm in dimension 2. For all results concerning lattice theory see [MG02], [SF]. At first, compute a reduced basis (U, V) of the lattice $L_{a,N} = \{(x, y) \in \mathbb{Z}^2 | ax = y \bmod N\}$ using the Gaussian algorithm. As we can easily find a sufficiently short basis of $L_{a,N}$, for example take the vectors $(1, a)$ and $(1, a + N)$, this can be done in time $\mathcal{O}((\log N)^3)$. Let T be a small solution and T_0 be any solution of (2). To find $T_0 = (x_0, y_0)$, we can choose x_0 as we like and then compute $y_0 = ax_0 - c \bmod N$. Define $l = 2^{k_0+2}$ and assume that $L_{a,N}$ is a so called l-*good* lattice, meaning that there exists no non-zero lattice vector shorter than l. This choice of l together with the properties of a reduced basis guarantee two important facts: in the first place, T is unique as a small solution of (2). Secondly, the coefficients of T in the basis (U, V) are smaller than $1/2$ in absolute value. Thus, the coefficients (in (U, V)) of the lattice point $T - T_0$ can be constructed simply by taking the closest integers to the coefficients of $-T_0$. This is a consequence of the uniqueness of basis representation. From knowledge of T_0 and $T - T_0$, we can easily construct T.

But as stated above, this method only works if the randomly chosen a yields an l-good lattice. We already have seen that the absolute number of bad values for a can be bounded above by 2^{2k_0+6}, consequently the probability of choosing a bad value is smaller than 2^{2k_0+6-k}. The total advantage of this reduction is therefore greater than $\varepsilon' = \varepsilon(\varepsilon - 2^{2k_0+6-k})$, where ε denotes the advantage of the partial inverter $\mathcal{A}$. Note that ε' is non-negligible in $k = \log N$, if ε is non-negligible in k and if k_0 is adequate smaller than k, i.e. there is a rational number $0 < t < 1/2$ such that $k_0 < tk$.

2.2 RSA-Paillier

Let N, e be the RSA public-key. The Hensel lifting problem of the RSA encryption function is to compute $r^e \bmod N^2$ for a given ciphertext $r^e \bmod N$. In 2002, Sakurai and Takagi proved that RSA-Paillier is one-way iff the Hensel-lifting problem is hard [ST02]. Moreover, they introduced a reduction algorithm for solving the RSA problem using the Hensel-lifting oracle as a subroutine. But this algorithm was not efficient (i.e. for each bit of the secret message two oracle-calls were needed), and it could be proven to achieve a non-negligible advantage only in case of a perfect Hensel-lifting oracle. A short time later, Catalano et al. were able to show that the RSA problem could be solved by calling the (potentially non-perfect) Hensel-lifting oracle only twice [CNS02], hence they reduced the one-wayness of RSA-Paillier to the RSA problem. We shortly explain their technique in the following.

Assume that a random RSA ciphertext $c = r^e \bmod N$ is given. We construct an algorithm that computes r given c, N, e using the Hensel lifting. The algorithm obtains $r^e \bmod N^2$ by invoking the Hensel lifting oracle. Then it computes $a^e r^e \bmod N$ for randomly chosen integer $a \in (\mathbb{Z}/N\mathbb{Z})^\times$, and obtains $\mu^e \bmod N^2$ from the Hensel lifting oracle, where $\mu = ar \bmod N$. There is an integer z such that $ar = \mu(1 + zN) \bmod N^2$. The integer $z \bmod N$ can be computed due to $a^e r^e = \mu^e(1 + ezN) \bmod N^2$. Consider the two-dimensional lattice $L = \{(R, U) \in \mathbb{Z}^2 | aR = U(1+zN) \bmod N^2\}$. By the lattice reduction algorithm we can find a vector $(r', \mu') \in L \cap [1, \ldots, N-1]^2$ in polynomial time of $\log N$. As the sought-after vector (r, μ) is an element of L, too, we have the relationship $r'\mu = r\mu' \bmod N^2$. Moreover, due to the size constraints $0 < r, r', \mu, \mu' < N$ we conclude that in fact equality holds, i.e. $r'\mu = r\mu'$.
Thus, r and μ are multiples of $r'/\gcd(r', \mu')$ and $\mu'/\gcd(r', \mu')$, respectively, with a factor that is given by $\gcd(r, \mu)$. As with overwhelming probability this factor is sufficiently small, it can be found efficiently by an exhaustive search.

Catalano et al. showed that their method works in time polynomial in $\log N$ with a non-negligible advantage, but they gave no concrete bounds.

3 The Proposed Reduction Algorithm

Let N be a natural number, $0 < a < N$, $0 \leq c < N$, and $\gcd(a, N) = 1$. In this section we give the outline of the algorithm Lin_Cong for finding small solutions of the two-variable linear modular congruence

$$ax = y + c \bmod N. \tag{3}$$

To be more concrete, we introduce an algorithm for finding so-called *x-minimal* solutions of (3).

Definition 1. *The pair $(\hat{x}, \hat{y}), 0 \leq \hat{x} < N, 0 \leq \hat{y} < B$ is called a x-minimal solution of (3) with respect to the bound $B, 0 < B < N$, if $(\hat{x}, \hat{y})$ possesses the following properties:*

1. $a\hat{x} = \hat{y} + c \bmod N$.
2. *$\hat{x}$ fulfills the following minimality condition: If (x_{alt}, y_{alt}) is a solution of the congruence (3) where $0 \leq y_{alt} < B$ holds, then we have $\hat{x} \leq x_{alt}$.*

Note that due to the condition $\gcd(a, N) = 1$ for each B there is exactly one x-minimal solution of (3) w.r.t. B.

As a second step, we propose an efficient variant of the algorithm with complexity $\mathcal{O}((\log N)^3)$. One application of the new algorithm is to replace the lattice based methods used in the reduction proofs described above. Note that we always use $\{0, 1, \ldots, N-1\}$ as representatives for the residue classes modulo N. The outline of the proposed algorithm is as follows:

Lin_Cong (Outline)

Input: a, c, N, B, where $0 < a, B < N$, $0 \leq c < N$, and $\gcd(a, N) = 1$
Output: $\hat{x}, \hat{y}$ such that $a\hat{x} = \hat{y} + c \bmod N$ and $\hat{x} \geq 0$ is minimal
with respect to the property that $0 \leq \hat{y} < B$

1. set $a' = a, c' = c, N' = N$
2. set $y' = -c' \bmod N'$
3. while $y' \geq B$ do
4. set $(a', N') = (-N' \bmod a', a')$ (parallel assignment)
5. set $c' = c' \bmod N', y' = -c' \bmod N'$
6. set $\hat{y} = y', \hat{x} = a^{-1} \cdot (\hat{y} + c) \bmod N$
7. return $(\hat{x}, \hat{y})$

In the following, we describe the idea of the proposed algorithm.
First note that $\gcd(a', N') = \gcd(a, N) = 1$ and $a' < N'$ holds in any iteration. Therefore we see that $a' = 0$ is only possible if the corresponding N' (the old value a') equals 1. If this is the case, in step 5 of this iteration we compute $y' = 0$ and the algorithm will terminate. Consequently, the assertion $a' = -N' \bmod a'$ is always defined.
Let $(\hat{x}, \hat{y})$ be the unique x-minimal solution of (3) w.r.t. B. We show that the algorithm Lin_Cong (Outline) on the inputs a, c, N, B returns $(\hat{x}, \hat{y})$. To be more precise, the algorithm finds $\hat{y}$ and then computes the corresponding $\hat{x} = a^{-1} \cdot (\hat{y} + c) \bmod N$. The main idea of the algorithm is to reduce the original problem to a smaller instance and iterating this process. This is done as follows: From $a\hat{x} = \hat{y} + c \bmod N$ we deduce $a\hat{x} = \hat{y} + c + kN$ for a suitable $k \in \mathbb{Z}$. Euclidian division yields $N = aq + r$ with $0 \leq r < a$ and a positive integer q. Hence we have

$$\begin{aligned} a\hat{x} = \hat{y} + c + kN = \hat{y} + c + k(aq + r) &\Rightarrow -rk = \hat{y} + c + a(kq - \hat{x}) \\ &\Rightarrow -rk = \hat{y} + c \bmod a \end{aligned}$$

Therefore we have constructed a new linear modular congruence with the new module a in the role of N and the new factor $-r = -N \bmod a$ in the role of a. A solution of this new congruence is given by $(k, \hat{y}) = (\frac{a\hat{x} - \hat{y} - c}{N}, \hat{y})$. The crucial point is the fact that this solution is the x-minimal solution w.r.t. B of the new

congruence.
We define the following sequences by iterating this process:

$$\begin{array}{llll} N_0 = N & a_0 = a & c_0 = c & x_0 = \hat{x} \\ N_{i+1} = a_i & a_{i+1} = -N_i \bmod a_i & c_{i+1} = c_i \bmod N_{i+1} & x_{i+1} = \frac{a_i x_i - \hat{y} - c_i}{N_i} \end{array}$$

Note that the first three columns exactly describe the corresponding sequences produced by the algorithm Lin_Cong (Outline). For this reason, we denote by $f_{\text{Lin\_Cong}}$ the transformation $(N_i, a_i, c_i) \mapsto (N_{i+1}, a_{i+1}, c_{i+1})$. Let us write cong_i for the linear modular congruence defined with the parameters a_i, c_i and N_i. Inductively, we conclude that the value x_i occurring in the last column leads to a solution $(x_i, \hat{y})$ of cong_i. Moreover, we can deduce the following lemma (for the rather technical proof see Appendix A):

Lemma 1. *Let $(x_i, \hat{y})$ be the x-minimal solution of cong_i w.r.t. B and let $x_i > 0$. Then $(x_{i+1}, \hat{y})$ is the x-minimal solution of cong_{i+1} w.r.t. B. In particular, the y-value of the current x-minimal solution w.r.t. B does not change during the transformation f_{Lin_Cong}, as long as x_i is non-negative.*

Hence with each iteration of the while loop the transformation $f_{\text{Lin\_Cong}}$ constructs a smaller problem, because the sequence of the moduli N_i is strictly monotone decreasing. The problem of finding the x-minimal solution is trivial in the following case:

Definition 2. *Let a, c, N, B be integers, where $0 < a, B < N$, $0 \leq c < N$, and $gcd(a, N) = 1$ hold. The congruence $ax = y + c \bmod N$ satisfies the zero-minimum condition with respect to B, if $-c \bmod N < B$ holds.*

In fact, it is an easy observation that the x-minimal solution of the congruence $ax = y + c \bmod N$ w.r.t. B is given by the pair $(0, -c \bmod N)$ iff $ax = y + c \bmod N$ satisfies the zero-minimum condition w.r.t. B. The aim of the algorithm Lin_Cong (Outline) is to convert the original congruence into a congruence satisfying the zero-minimum condition w.r.t. B. This is done using the transformation $f_{\text{Lin\_Cong}}$, which does not affect the y-value of the current x-minimal solution w.r.t. B.

Indeed, we can prove the correctness of algorithm Lin_Cong (Outline):

Theorem 1. *Algorithm Lin_Cong (Outline) is correct, i.e. given integers a, c, N, B, where $0 < a, B < N$, $0 \leq c < N$, and $gcd(a, N) = 1$ holds, the algorithm terminates and outputs the unique x-minimal solution $\hat{x}, \hat{y}$ of the congruence $ax = y + c \bmod N$ with respect to the bound B (see Definition 1).*

Proof. Let y_i denote the y-value computed by the algorithm Lin_Cong (Outline) in the ith iteration of the while loop. Note that per definition this value yields the solution $(0, y_i)$ of cong_i. For each $i = 0, 1, 2, \ldots$ the following holds: Either cong_i satisfies the zero-minimum condition w.r.t. B and consequently $(0, y_i)$ is the x-minimal solution of cong_i w.r.t. B. Or x_i, the x-value of the x-minimal solution of cong_i, is greater zero and lemma 1 tells us that $(x_{i+1}, \hat{y})$ equals the

x-minimal solution of cong_{i+1} w.r.t. B. As the sequence of the moduli N_i is strictly monotone decreasing, there must be an $i \geq 0$ such that cong_i satisfies the zero-minimum condition w.r.t. B. If this iteration is reached (i.e. we have $y_i < B$ for the first time), then $(0, y_i) = (x_i, \hat{y})$ must hold because according to lemma 1 we know that the y-value of the x-minimal solution w.r.t. B has not changed. Obviously, the x-value computed in step 6 is the correct one.

Analyzing algorithm Lin_Cong (Outline) we see that the parallel assignment in step 4 describes a variant of the Euclidian algorithm (set $(a, b) = (-b \bmod a, a)$ instead of set $(a, b) = (b \bmod a, a)$ for $a \leq b$). Obviously, the result remains the same, but unfortunately the variant is less efficient. In particular, in the worst case we need $a - 1$ steps (to see this, try $a = b - 1$), which is by far not fast enough. But some modifications may be helpful: A closer look at the recursion formula $(a, b) = (-b \bmod a, a)$ discloses, that problems occur if $b - a << a$ holds. In the following steps the difference $b - a$ is subtracted from a and b until the resulting a is smaller than $b - a$. This procedure may take a long (too long) time. Its result will be $(a \bmod (b - a), b - k(b - a))$, where k equals $a \div (b - a)$[3]. Therefore we gain a notable speedup by the following case differentiation:

if $b - a \geq a$ then set $(a, b) = (-b \bmod a, a)$
else set $(a, b) = (a \bmod b - a, b - k(b - a))$ with $k = a \div (b - a)$.

But we need to be a little careful if we wish to assign this idea to the original algorithm (with a' in the role of a and N' in the role of b). In detail, we must not ignore a reduction of the value c' which would have occurred in one of the skipped steps. A possible way out is to skip fewer steps, i.e. we subtract $N' - a'$ until the resulting a' is smaller than $N' - a'$ or c' is greater than the resulting N'. We will see in a while that these modifications are good enough to yield a polynomial running time (in $\log N$). But before doing so, we have to face a last problem: It is possible that the value $\hat{y}$ we are seeking for would be computed in one of the skipped steps. Note that in each skipped step the value $y' = -c' \bmod N'$ is reduced by the amount $N' - a'$ (this is true because due to the above considerations the value c' remains constant). Hence if the resulting y' exceeds the bound B, all the "invisible" values y' computed during the skipped steps do so, too. This means that it is possible to miss the sought-after value $\hat{y}$ only in the last while cycle before termination. So we avoid missing the correct $\hat{y}$ by doing the following: If steps have been skipped during the last while cycle add $N' - a'$ to the current value y' until $y' + k(N' - a')$ exceeds B for the first time. Then set $\hat{y} = y' + (k - 1)(N' - a')$ and compute the corresponding $\hat{x}$-value as usual.

3.1 Algorithm Lin_Cong

The proposed algorithm Lin_Cong is as follows:

[3] $x \div y$ denotes the Euclidian quotient of x and y

Lin_Cong

Input: a, c, N, B, where $0 < a, B < N$, $0 \le c < N$, and $\gcd(a, N) = 1$
Output: $\hat{x}, \hat{y}$ such that $a\hat{x} = \hat{y} + c \bmod N$ and $\hat{x} \ge 0$ is minimal
with respect to the property that $0 \le \hat{y} < B$

1. set $a' = a, c' = c, N' = N$
2. set $y' = -c' \bmod N'$
3. while $y' \ge B$ do
4. set diff$=N' - a'$
5. if diff $< a'$ and diff $< N' - c'$
6. then set $k = \min(a' \div \text{diff}, (N' - c') \div \text{diff})$
7. set $(a', N') = (a' - k \cdot \text{diff}, N' - k \cdot \text{diff})$, set flag $= 1$
8. else set $(a', N') = (-N' \bmod a', a')$, set flag $= 0$
9. set $c' = c' \bmod N'$, set $y' = -c' \bmod N'$
10. If flag $= 1$ then set $k = \left\lceil \frac{B - y'}{\text{diff}} \right\rceil - 1$, set $\hat{y} = y' + k \cdot \text{diff}$
11. else set $\hat{y} = y'$
12. set $\hat{x} = a^{-1} \cdot (\hat{y} + c) \bmod N$
13. return $(\hat{x}, \hat{y})$

We can prove the following theorem:

Theorem 2. *a) The complexity of the algorithm Lin_Cong is* $\mathcal{O}((\log N)^3)$.
b) Let a, c, N, B be integers, where $0 < a, B < N$, $0 \le c < N$, and $\gcd(a, n) = 1$ holds. Algorithm Lin_Cong on the inputs a, c, N, B finds a small solution $0 \le \hat{x}, \hat{y} < B$ of the congruence $ax = y + c \bmod N$, provided such a solution exists at all.

Proof. From the discussion above we know that on each input the algorithm Lin_Cong computes the same output as its slower variant Lin_Cong (Outline). Thus the second part of the theorem is an immediate consequence of theorem 1. So it remains to show that algorithm Lin_Cong runs in polynomial time. We distinguish four cases

1. The condition in step 5 is not fulfilled due to $N' - a' = \text{diff} \ge a'$. Hence the else-case in step 8 is entered. From $N' \ge 2a'$ we deduce that the assignment $N' = a'$ at least halves the value of N'.
2. The condition in step 5 is not fulfilled due to $N' - a' = \text{diff} \ge N' - c'$. Hence the else-case in step 8 is entered and N' is assigned to a'. Because of $a' \le c'$ the reduction of c' modulo $N' (= a')$ in step 9 at least halves the value of c'.
3. The condition in step 5 is fulfilled and the value k computed in step 6 equals $a' \div (N' - a')$. In this case, the assignment $a' = a' - k \cdot (N' - a')$ done in step 7 is equivalent to $a' = a' \bmod (N' - a')$. As we have $N' - a' < a'$, this assignment at least halves the value of a'.
4. The condition in step 5 is fulfilled and the value k computed in step 6 equals $(N' - c) \div (N' - a')$. The value of k is chosen in order to achieve that $N' - (k+1)(N' - a') \le c'$ holds. Hence the reduction of c' modulo N' in step 9 of the following while cycle at least halves the value of c'.

Summing up, we see that at least in each second while cycle at least one of the values a', c' and N' is at least halved. Note that the algorithm terminates at once if $a' = 0, c' = 0$ or $N' = 1$ holds. So the number of while cycles is bounded above by $\log a + 2\log c + \log N$. Each step during the while loop can be done in $\mathcal{O}((\log N)^2)$, therefore the time complexity of the algorithm Lin_Cong is $\mathcal{O}((\log N)^3)$.

3.2 Finding All Small Solutions

In this subsection we show that algorithm Lin_Cong can be modified to find all small solutions (x, y) of the linear modular congruence (3). We call (x, y) a *small* solution, if $0 \leq x, y < \sqrt{N}$ is satisfied. The time needed for computing all small solutions is $\mathcal{O}((\log N)^3) + l\mathcal{O}(\log N)$, where l is the absolute number of small solutions. The most important observation is that there is a quite simple relationship between all the small solutions in the case of $c = 0$. The general case $c \neq 0$ can be easily derived from the special case. We will see that in both cases all small solutions are located on the same line.

The Case $c = 0$ Let (x_0, y_0) and (x_1, y_1) be two different small solutions of

$$ax = y \bmod N, \ \gcd(a, N) = 1, \tag{4}$$

i.e. we have

$$ax_0 = y_0 \bmod N \text{ and } ax_1 = y_1 \bmod N,$$

leading to

$$x_0 y_1 = x_1 y_0 \bmod N.$$

But due to the size-constraints we deduce that this relationship even holds in $\mathbb{Z}$. Consequently, all small solutions are located on the same line through the origin. Hence to get all small solutions we simply have to compute all integer multiples $(k\hat{x}, k\hat{y}), k \in \mathbb{Z}^{\geq 0}, k\hat{x}, k\hat{y} < \sqrt{N}$, where $(\hat{x}, \hat{y})$ is the smallest non-zero solution of (4). This solution can be obtained using algorithm Lin_Cong. If we run algorithm Lin_Cong on an input with $c = 0$, then it will terminate at once with the result $(0, 0)$. But we are seeking for a non-zero solution, hence we exploit the relationship $ax = y \bmod N \Leftrightarrow a(x-1) = y - a \bmod N$. Namely, we run Lin_Cong on the input $(a, N-a, N, \sqrt{N})$, get the result (x', y'), and return $(\hat{x}, \hat{y}) := (x' + 1, y')$. Theorem 1 shows that $(\hat{x}, \hat{y})$ indeed yields the smallest non-zero solution of (4).

The Case $c \neq 0$ Let $(\hat{x}, \hat{y})$ be the small solution computed by algorithm Lin_Cong on the input $(a, c, N, \sqrt{N})$ and let (x_{alt}, y_{alt}) be a different small solution. In particular, the difference $(x_{alt} - \hat{x}, y_{alt} - \hat{y})$ is a non-zero solution of (4). As $\hat{x}$ is minimal, we know $\hat{x} < x_{alt}$. Thus we conclude $0 < x_{alt} - \hat{x} < \sqrt{N}, -\sqrt{N} < y_{alt} - \hat{y} < \sqrt{N}$. We distinguish two cases:

1. If $y_{alt} > \hat{y}$ holds, then $(x_{alt} - \hat{x}, y_{alt} - \hat{y})$ is a small solution of (4) and can be found as described above.
2. Otherwise $(x_{alt} - \hat{x}, y_{alt} - \hat{y})$ is a solution of (4), too, but only small in absolute value (with a negative y-component). It is easy to see that we can find all solutions $(x, y), 0 \leq x < \sqrt{N}, -\sqrt{N} < y \leq 0$ of (4) by computing all small solutions of $(-a)x = y \bmod N$ as usual and then changing the signs of the y-components.

Note that at most one of these two cases may appear, because if there are two additional small solutions (x_{alt1}, y_{alt1}) and (x_{alt2}, y_{alt2}) with $y_{alt1} > \hat{y}$ and $y_{alt2} < \hat{y}$, then the three differences $(x_{alt1} - \hat{x}, y_{alt1} - \hat{y})$, $(x_{alt2} - \hat{x}, y_{alt2} - \hat{y})$ and $(x_{alt1} - x_{alt2}, y_{alt1} - y_{alt2})$ must be located on at most two lines through the origin, a contradiction.
This leads to the following algorithm:

Lin_Cong_All

Input: a, c, N, where $0 < a, c < N$, and $\gcd(a, N) = 1$
Output: Set $S = \{(x, y) | ax = y + c \bmod N, 0 \leq x, y < \sqrt{N}\}$

1. set $S = \{\}$.
2. set $(\hat{x}, \hat{y}) =$ Lin_Cong$(a, c, N, \sqrt{N})$
3. if $\hat{x} \geq \sqrt{N}$ then return S and stop
4. else append $(\hat{x}, \hat{y})$ to S
5. set $(x', y') =$ Lin_Cong$(a, N - a, N, \sqrt{N})$
6. set $(x_0, y_0) = (x' + 1, y')$, set $k = 1$
7. while $\hat{x} + kx_0 < \sqrt{N}$ and $\hat{y} + ky_0 < \sqrt{N}$ do
8. append $(\hat{x} + kx_0, \hat{y} + ky_0)$ to S and increment k
9. if $\#S > 1$ then return S and stop
10. set $(x', y') =$ Lin_Cong$(N - a, a, N, \sqrt{N})$
11. set $(x_0, y_0) = (x' + 1, -y')$, set $k = 1$
12. while $\hat{x} + kx_0 < \sqrt{N}$ and $\hat{y} + ky_0 \geq 0$ do
13. append $(\hat{x} + kx_0, \hat{y} + ky_0)$ to S and increment k
14. return S

In step 2, we use algorithm Lin_Cong to compute the small solution with the minimal x-coordinate $\hat{x}$. If even $\hat{x}$ exceeds the bound $\sqrt{N}$, then obviously no small solution exists at all. The value (x_0, y_0) computed in step 6 equals the smallest non-zero solution of (4). As we have seen above, each sum of $(\hat{x}, \hat{y})$ and an integer multiple of (x_0, y_0) yields a solution of $ax = y + c \bmod N$. But as $\hat{x}$ is minimal, we only have to consider factors $k \geq 1$. If there is at least one small solution $(\hat{x} + kx_0, \hat{y} + ky_0)$, we know that all small solutions have to be of this shape. Hence the while loop in step 7 and 8 has finds all remaining small solutions and the algorithm terminates. Otherwise we compute the smallest non-zero solution of $ax = y \bmod N$ with a negative y-component (step 10 and 11) and proceed in the same way as before.

3.3 Comparison with the Continuous Fraction Method

Another often used method for finding small solutions of linear modular congruence where the affine coefficient c equals zero is obtained by the continued fraction expansion. We call this method the Euclidean reduction (See [HW79] for the comprehensive treatment). To resume, this method finds all fractions $\frac{p}{q}$ nearby a rational number α (i.e. we have $|\alpha - \frac{p}{q}| < \frac{1}{2q^2}$), where the fractions $\frac{p}{q}$ come in their lowest terms. Assume that we want to find small solutions that do not exceed $\sqrt{N}$ of the congruence $ax = y \bmod N$, where $\gcd(a, N) = 1$ holds. As we have already shown in subsection 3.2, all these solutions are located on the same line through the origin. Therefore there exists a solution $(\hat{x}, \hat{y})$ such that $\gcd(\hat{x}, \hat{y}) = 1$ is fulfilled. From $a\hat{x} = \hat{y} \bmod N$ we conclude that there is an integer k such that $a\hat{x} = \hat{y} + kN$. We have

$$\frac{a}{N} - \frac{k}{\hat{x}} = \frac{\hat{y}}{N\hat{x}}. \tag{5}$$

If $2\hat{x}\hat{y} < N$ holds, the upper-bound of $\hat{y}/N\hat{x}$ is $1/2\hat{x}^2$. If in addition the rational number $k/\hat{x}$ is irreducible, i.e. $\gcd(k, \hat{x}) = 1$, we can find the integer $\hat{x}$ and thus $\hat{y}$ by using the Euclidian reduction method. Note that $\gcd(k, \hat{x}) = 1$ holds because $\gcd(\hat{x}, \hat{y}) = 1$ is satisfied. If $\gcd(k, \hat{x}) = 1$ is not true, there is an integer $\delta > 1$ such that $\gcd(k, \hat{x}) = \delta$. From $a\hat{x} - Nk = \hat{y}$, we have $\delta|\hat{y}$ and hence $\delta|\gcd(\hat{x}, \hat{y})$. It contradicts to $\gcd(\hat{x}, \hat{y}) = 1$.
Summing up, we can use this method if we know that the product $2\hat{x}\hat{y}$ does not exceed N. Consequently, we prefer the use of algorithm Lin_Cong, which finds $\hat{x}, \hat{y}$, even if $2\hat{x}\hat{y} < N$ is not fulfilled.

4 Security Reduction Analysis using the Proposed Algorithm

In this section, we show how algorithm Lin_Cong may be applied to the reduction proofs of RSA-OAEP and RSA-Paillier. In the case of RSA-OAEP we will upper-bound the number of bad values a by 2^{2k_0+1}, compared to the former bound 2^{2k_0+6}. Regarding to RSA-Paillier, we will give an explicit reduction algorithm based on the work of Catalano et al. [CNS02]. We will achieve reduction time $2t + \mathcal{O}((\log N)^3\varepsilon^{-2})$ and advantage $\varepsilon' > \varepsilon^2/5$ where t and ε are the time and the advantage of the Hensel-lifting oracle, respectively.

4.1 Application to RSA-OAEP

In section 2.1 we have described the reduction proof given by Fujisaki et al. [FOPS01]. Remember that they have constructed the following congruence

$$ax = y + c \bmod N, \;\; c = (v - ua) \cdot 2^{k_0} \bmod N, \tag{6}$$

where u and v are built of the $k - k_0$ most significant bits of m or $ma \bmod N$, respectively. In the RSA-OAEP case we call (x, y) a *small* solution of the congruence (6) iff $0 \leq x, y < 2^{k_0}$ holds. The congruence (6) is known to have the small solution (r, s), where r is built from the remaining k_0 least significant bits of m.

In section 2.1 we have already seen that the lattice based method only works if the randomly chosen value a yields an l- good lattice. In contrast, algorithm Lin_Cong always finds a small solution, provided a small solution exists at all. But it has to be stressed, that referring to the lattice method the choice of a good value a ensures that there exists exactly one small solution. This is an important property, because if the small solution (r, s) is not unique, there is of course no warrant that the solution computed with our algorithm is the correct one. A possible way out is to use algorithm Lin_Cong_All instead, which computes all small solutions, and to test each of them. But this is only efficient if the set of small solutions is not too big. Let l be a "sufficiently" small natural number. We want to bound above the probability that the number of small solutions does not exceed l. As we have seen in subsection 3.2, each small solution is of the shape $(\hat{x}+kx_0, \hat{y}+ky_0)$, where $(\hat{x}, \hat{y})$ is the special solution computed by the algorithm Lin_Cong and (x_0, y_0) is either the shortest element of $\{(x, y)|ax = y \bmod N, 0 < x, y < 2^{k_0}\}$ or (x_0, y_0) is the shortest element of $\{(x, y)|ax = y \bmod N, 0 < x < 2^{k_0}, -2^{k_0} < y < 0\}$. Hence there are at most l small solutions of (6), iff the congruence $ax = y \bmod N$ has no solution (x, y), where

$$0 < x < 2^{k_0}/l, -2^{k_0}/l < y < 2^{k_0}/l. \tag{7}$$

We call a a *bad value*, if there exists a solution of $ax = y \bmod N$ fulfilling the size constraints (7). If (7) holds for (x, y), then there is exactly one a such that (x, y) is a solution of $ax = y \bmod N$, namely $a = x^{-1}y \bmod N$. Note that due to the size constraints no problems of computing modular inverses occur. Hence there are at most $2^{2k_0+1}/l^2$ bad values of a. The maximal number is 2^{2k_0+1} for $l = 1$.

Therefore, in case of using the lattice solution the probability to choose a bad value a is at least 2^5 times greater compared with the corresponding probability in case of using algorithm Lin_Cong_All. We finish with the following theorem:

Theorem 3. *Assume there is an adversary that on input $N, e, m^e \bmod N$ returns the $k - k_0$ most significant bits of m with advantage ε and in time t, where N is a k-bit RSA modulus, e is a public key belonging to N and $2k_0 < k$ holds. Let $l \leq (\log N)^2$ be any natural number. Then with advantage $\varepsilon' > \varepsilon(\varepsilon - 2^{2k_0+1-k}/l^2)$ and in time $2t + \mathcal{O}((\log N)^3)$ we can compute a set S with $m \in S$ and $\#S \leq l$.*

If we set $l = 1$ we get the following corollary:

Corollary 1 *Assume there is an adversary that on input $N, e, m^e \bmod N$ returns the $k - k_0$ most significant bits of m with advantage ε and in time t, where N is a k-bit RSA modulus, e is a public key belonging to N and $2k_0 < k$ holds. Then we can break the RSA problem related to (N, e) with advantage at least $\varepsilon(\varepsilon - 2^{2k_0+1-k})$ and in time $2t + \mathcal{O}((\log N)^3)$.*

Note that this achievement is the more valuable the smaller the difference $k - 2k_0$ is. However, in the case of PKCS #1 v2.0, k_0 is much smaller than $k/2$, therefore in this case the result is rather of theoretical nature.

4.2 Application to RSA-Paillier Cryptosystem

In section 2.2 we have described the reduction proof given by Catalano et al. [CNS02]. Remember that they have constructed the following congruence

$$Ax = y \bmod N^2,\ A = a(1 + zN)^{-1} = a(1 - zN) \bmod N^2, \tag{8}$$

which is known to have the solution (r, μ), where r, μ are elements of $\mathbb{Z}/N\mathbb{Z}$ and r is the sought-after RSA message. Hence (r, μ) is a small solution of (8) as described in subsection 3.2, where we have seen how to find all small solutions. To be concrete, $\mathsf{Lin\_Cong}$ on the input $(A, N^2 - A, N^2, N)$ finds the smallest non-zero solution of (8) and all other small solutions come as integer multiples of this special solution.

We describe the explicit reduction algorithm as follows:

OW_RSA_Paillier
Input: (N, e) RSA Public-key, c ciphertext, $\mathcal{O}_{RSAP}$ Hensel-Lifting oracle
Output: Message r such that $c = r^e \bmod N$ or an integer divisor of r
1. obtain $t = \mathcal{O}_{RSAP}(c)$
2. generate random $a \in (\mathbb{Z}/N\mathbb{Z})^\times$
3. obtain $s = \mathcal{O}_{RSAP}(a^e c \bmod N)$
4. compute $v = ta^e s^{-1} \bmod N^2$
5. compute $z = \frac{(v-1)}{N} e^{-1} \bmod N$
6. compute $A = a(1 - zN) \bmod N^2$
7. compute $(\hat{x}, \hat{y}) = \mathsf{Lin\_Cong}(A, N^2 - A, N^2, N)$
8. return $\hat{x} + 1$

Obviously, the running time of this algorithm is $\mathcal{O}((\log N)^3)$ plus the time needed for calling the Hensel-Lifting oracle twice. To receive the original value r, we have to test if $(kr)^e = c \bmod N$, where the multiplier k runs from 1 to (the unknown number) $\gcd(r, \mu)$. In the following, we upper-bound the probability that $\gcd(r, \mu)$ is not sufficiently small. We exploit the following estimate (see [NZM91]):

$$\frac{\pi^2}{3}\left(\frac{2N^2 - 2N}{4N^2 + 4N + 1}\right) < \sum_{i=1}^{N} \frac{1}{i^2} < \frac{\pi^2}{3}\left(\frac{2N^2 + 2N}{4N^2 + 4N + 1}\right).$$

Hence we have

$$\begin{aligned}
\#\{(a,b) \in [1,\ldots,N]^2 | \gcd(a,b) > B\} &< \sum_{i=B+1}^{N} \frac{N^2}{i^2} \\
&< \frac{N^2\pi^2}{3}\left(\frac{2N^2+2N}{4N^2+4N+1} - \frac{2B^2-2B}{4B^2+4B+1}\right) \\
&< \frac{N^2\pi^2}{3}\left(\frac{1}{2} - \frac{2B^2-2B}{4B^2+4B+1}\right).
\end{aligned}$$

As a simple computation shows that

$$\frac{1}{2} - \frac{2B^2-2B}{4B^2+4B+1} < \frac{1}{B},$$

we finally conclude

$$\#\{(a,b) \in [1,\ldots,N]^2 | \gcd(a,b) > B\} < \frac{4N^2}{B}.$$

The values r and μ are independently chosen and uniformly distributed elements of $\mathbb{Z}/N\mathbb{Z}$. Replacing $5\varepsilon^{-2}$ for B, we therefore deduce that the probability that $\gcd(r,\mu)$ exceeds $5\varepsilon^{-2}$ is bounded above by $4\varepsilon^2/5$.

This leads to the following theorem:

Theorem 4. *Let $\mathcal{O}_{RSAP}$ be the Hensel-lifting oracle that computes $r^e \bmod N^2$ for given $r^e \bmod N$ with advantage ε and in time t. Using $\mathcal{O}_{RSAP}$ as a subroutine, we can break the RSA problem (N,e) with advantage $\varepsilon' > \varepsilon^2/5$ and in time $2t + \mathcal{O}((\log N)^3\varepsilon^{-2})$.*

An Example of OW_RSA_Paillier We present a small example of reduction algorithm OW_RSA_Paillier. We choose the public-key of the RSA-Paillier cryptosystem as $(e,N) = (7, 9359629)$. In our case N^2 is equal to 87602655017641. Let $c = 2592708$ be the target ciphertext. We intend to find the integer r such that $c = r^e \bmod N$ using the oracle $\mathcal{O}_{RSAP}$.

In step 1 we ask c to oracle $\mathcal{O}_{RSAP}$, and we obtain $t = \mathcal{O}_{RSAP}(c) =$ 37278188147938. In step 2, a random integer $a \in (\mathbb{Z}/N\mathbb{Z})^\times$ is generated, and we choose $a = 5973500$. In step 3, we compute $\mu^e = a^e c \bmod N$, ask it to oracle $\mathcal{O}_{RSAP}(\mu^e \bmod N)$, then we obtain $\mu^e \bmod N^2 = 59913274976876$. In step 4 and 5, integer z such that $ar = \mu(1+zN) \bmod N^2$ is computed, and in our case $z = 9040417$. In step we obtain the linear equation $Ar = \mu \bmod N^2$ for $A = 35049167803493$ and two unknown variables $0 < r, \mu < N$.

In the following, we solve this linear equation using algorithm Lin_Cong. We list up the intermediate values of N', a', c' and y', where N', a' and c' are initialized with N^2, A and $N^2 - A$. The while loop terminates if $y' < N$ holds. Step 10 and 11 of Lin_Cong are dedicated to compute the output values $\hat{x}$ and $\hat{y}$, which in our case equal $r - 1$ and μ.

N'	a'	c'	y'	
87602655017641	35049167803493	52553487214148	35049167803493	
35049167803493	17544848392838	17504319410655	17544848392838	
17544848392838	40528982183	17504319410655	40528982183	
40528982183	4200892401	36328089782	4200892401	
4200892401	1479941827	2720950574	1479941827	
1479941827	238933080	1241008747	238933080	
238933080	192589733	46343347	192589733	
53559692	7216345	46343347	7216345	← loop exit

The output values are $(1835097, 7216345) = (r - 1, \mu)$. Namely, we successfully find $r = 1835098$.

5 Conclusion

In this paper we investigated several security reduction algorithms related to RSA-OAEP and RSA-Paillier cryptosystems. These algorithms require to solve a linear modular equation with a small solution The standard algorithms for solving this task are Gaussian reduction and Euclidean reduction. We proposed an efficient alternative algorithm and showed its preferences. In the case of RSA-OAEP we were able to enhance the advantage of the reduction proof. Referring to RSA-Paillier, the use of our new algorithm provides us the complete security reduction proof, including explicit bounds for time costs and the achieved advantage.

References

[BR95] M. Bellare and P. Rogaway. Optimal Asymmetric Encryption - how to encrypt with RSA. In *Advances in Cryptology - EUROCRYPT 84*, volume 950 of *Lecture Notes in Computer Science*, pages 92–111. Springer-Verlag, 1995.

[CGHGN01] D. Catalano, R. Gennaro, N. Howgrave-Graham, and P. Nguyen. Paillier's cryptosystem revisited. In *The 8th ACM conference on Computer and Communication Security*, pages 206–214, 2001.

[CNS02] D. Catalano, P. Nguyen, and J. Stern. The hardness of Hensel lifting: The case of RSA and discrete logarithm. In *Advances in Cryptology - ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 299–310, Berlin, 2002. Springer-Verlag.

[FOPS01] E. Fujisaki, T. Okamoto, D. Pointcheval, and J. Stern. RSA-OAEP is secure under the RSA assumption. In *Advances in Cryptology - CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 260–274, 2001.

[GM97] M. Girault and J.-F. Misarsky. Selective forgery of RSA signatures using redundancy. In *Advances in Cryptology - EUROCRYPT 97*, volume 1233 of *Lecture Notes in Computer Science*, pages 495–507. Springer-Verlag, 1997.

[HW79] G. Hardy and E. Wright. *An Introduction to The Theory Of Numbers*. Oxford Press, fifth edition edition, 1979.

[JC86] W. De Jonge and D. Chaum. Attacks on some RSA signatures. In *Advances in Cryptology - CRYPTO 85*, volume 218 of *Lecture Notes in Computer Science*, pages 18–27. Springer-Verlag, 1986.

[Kat01] S. Katzenbeisser. *Recent Advances in RSA Cryptography.* Kluwer Academic Publishers Group, 2001.

[MG02] D. Micciancio and S. Goldwasser. *Complexity of Lattice Problems – A Cryptographic Perspective.* Kluwer Academic Publishers Group, 2002.

[NZM91] I. Niven, H.S. Zuckerman, and H.L. Montgomery. *An Introduction to the Theory of Numbers.* John Wiley & Sons, Inc., 1991.

[SF] C. P. Schnorr and R. Fischlin. Gittertheorie und algorithmische Geometrie. available from `http://ismi.math.uni-frankfurt.de/schnorr/lecturenotes/gitter.pdf`.

[Sho02] V. Shoup. OAEP reconsidered. *Journal of Cryptology*, (15):223–249, 2002.

[ST02] K. Sakurai and T. Takagi. New semantically secure public-key cryptosystems from the RSA primitive. In *Public Key Cryptography, 5th International Workshop on Practice and Theory in Public Key Cryptosystems, PKC 2002*, volume 2274 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2002.

A Proof of Lemma 1

Before starting the remaining proof of lemma 1, we recapitulate the notations given in section 3: We write $(\hat{x}, \hat{y})$ for the unique x-minimal solution of the congruence $ax = y + c \bmod N$ w.r.t. the bound B. The variables y_i, a_i, N_i and c_i constitute the corresponding values produced by the algorithm $\mathsf{Lin\_Cong}$ $\mathsf{(Outline)}$ in the ith iteration of the while loop, and the value x_i is computed from $a_{i-1}, c_{i-1}, \hat{y}$ and N_{i-1} by $x_i = \frac{a_{i-1}x_{i-1} - \hat{y} - c_{i-1}}{N_{i-1}}$. The linear modular congruence $a_i x = y + c_i \bmod N_i$ is abbreviated by cong_i.

At first, we introduce a kind of "solution lifting":

Proposition 1 *Let (x, y) be a solution of $cong_{i+1}$. Then the pair $\left(\frac{y+c_i+xN_i}{a_i}, y\right)$ is a solution of $cong_i$. If in addition $0 \le x, y < N_{i+1}$ holds then $0 \le \frac{y+c_i+xN_i}{a_i} \le N_i$ is fulfilled.*

Proof. Note that the value $a_i = N_{i-1}$ cannot be zero, since otherwise iteration $(i+1)$ would not have been reached. First, we show $\frac{y+c_i+xN_i}{a_i} \in \mathbb{Z}$. The recursion formulas define $N_{i+1} = a_i$ and $c_{i+1} = c_i \bmod N_{i+1} = c_i \bmod a_i$. Hence there is an integer l such that c_{i+1} equals $c_i + la_i$. As (x, y) is a solution of cong_{i+1} we conclude

$$a_i \,|\, y + c_{i+1} - xa_{i+1} = y + (c_i + la_i) - x(-N_i \bmod a_i) \;\Rightarrow\; a_i \,|\, y + c_i + xN_i$$

It follows immediately that $\left(\frac{y+c_i+xN_i}{a_i}, y\right)$ is an integer solution of cong_i.

Now assume $0 \le x, y < N_{i+1} = a_i$. We have

$$y + c_i + xN_i \le y + c_i + (a_i - 1)N_i < a_i + N_i + (a_i - 1)N_i = a_i + a_iN_i$$

Therefore we conclude $\frac{y+c_i+xN_i}{a_i} < N_i + 1$, which finishes the proof.

Now we are prepared to prove lemma 1.

Lemma 1. *Let $(x_i, \hat{y})$ be the x-minimal solution of $cong_i$ w.r.t. B and let $x_i > 0$. Then $(x_{i+1}, \hat{y})$ is the x-minimal solution of $cong_{i+1}$ w.r.t. B. In particular, the y-value of the current x-minimal solution w.r.t. B does not change during the transformation f_{Lin_Cong}, as long as x_i is non-negative.*

Proof. Assume that (x_{alt}, y_{alt}) is a solution of cong_{i+1} where $0 \leq x_{alt} < N_{i+1}$ and $0 \leq y_{alt} < B$. Our goal is to show $x_{alt} \geq x_{i+1} \geq 0$.
First we prove that x_{i+1} is non-negative. Note that $a_i > 0$ must hold because otherwise the iteration $(i+1)$ of the while loop would not have been reached. From the definition of $x_{i+1} = \frac{a_i x_i - \hat{y} - c_i}{N_i}$ and the condition $x_i > 0$ we therefore conclude $\hat{y} + c_i + x_{i+1} N_i > 0$. Assume $x_{i+1} < 0$. Hence we have

$$\hat{y} + c_i > N_i \ \Rightarrow \ \hat{y} > N_i - c_i \geq -c_i \bmod N_i = y_i.$$

Thus $(0, y_i)$ is a solution of cong_i with $y_i < B$. This contradicts the preconditions, namely that $(x_i, \hat{y})$ is the x-minimal solution of cong_i w.r.t. B and $x_i > 0$. Consequently, we must have $x_{i+1} \geq 0$.

From proposition 1 we conclude that the pair

$$\left(\frac{y_{alt} + c_i + x_{alt} N_i}{a_i}, y_{alt} \right)$$

is a solution of cong_i, in particular we have $0 \leq \frac{y_{alt} + c_i + x_{alt} N_i}{a_i} \leq N_i$. As $(x_i, \hat{y})$ is the x-minimal solution of cong_i w.r.t. B, we conclude

$$\begin{aligned} \frac{y_{alt} + c_i + x_{alt} N_i}{a_i} - x_i \geq 0 & \\ \Rightarrow x_{alt} \geq \frac{a_i x_i - y_{alt} - c_i}{N_i}. & \qquad (9) \end{aligned}$$

In the case of $y_{alt} \leq \hat{y}$ this immediately leads to the desired result $x_{alt} \geq x_{i+1} = \frac{a_i x_i - \hat{y} - c_i}{N_i}$. Thus we consider the case $y_{alt} > \hat{y}$. Looking at the difference between x_{i+1} and the right side of (9) we observe

$$x_{i+1} - \frac{a_i x_i - y_{alt} - c_i}{N_i} = \frac{-\hat{y} + y_{alt}}{N_i} < \frac{B}{N_i} < 1. \qquad (10)$$

Note that the last inequality must hold since in the case of $N_i \leq B$ the iteration $(i+1)$ of the while loop would not have been reached. Therefore we finally conclude $x_{alt} \geq x_{i+1}$ from (9), (10), and the fact that both of x_{alt} and x_{i+1} are integers.