Computation of Approximate L-th Roots Modulo n and Application to Cryptography

Marc Girault
Service d'Etudes communes des Postes et Télécommunications
BP 6243, 14066 Caen Cedex, France.

Philippe Toffin, Brigitte Vallée
Département de mathématiques
Université, 14032 Caen Cedex, France.

ABSTRACT

The goal of this paper is to give a unified view of various known results (apparently unrelated) about numbers arising in crypto schemes as RSA, by considering them as variants of the computation of approximate L-th roots modulo n. Here one may be interested in a number whose L-th power is "close" to a given number, or in finding a number that is "close" to its exact L-th root. The paper collects numerous algorithms which solve problems of this type.

This work was supported by a convention between SEPT and University of Caen.

I. INTRODUCTION

That a lot of public-key cryptosystems or digital signature schemes are based on the computation of L-th roots modulo n is today a very well known fact. Roughly speaking, and assuming that n is a large integer (say, at least, a 320-bit long one), this computation is easy when n is prime or when all its prime factors are known, hard when n is composite and its factors unknown. The cryptographic validity of the famous system RSA [RSA] (as well as many other systems) is based on this dissymmetry.

But very often in public-key cryptology, the problem is raised of extracting *approximate* L-th roots modulo n, in a sense that will be stated more precisely in next section. As this problem is weaker than the problem of extracting *exact* L-th roots modulo n, we may hope that it will be solved even when the factors of n remain hidden. As shown below, that hope is often fulfilled, provided that we do not demand a "too good" approximation.

For example, in the Morrison-Brillhart factorization algorithm [MB], the most consuming part is the quest of integers x such that x^2 (mod n) is as small as possible (hoping that it is "smooth"), where n is the number to be factored. The continued fraction algorithm allows us to find such values of x, but most of the time, x^2 (mod n) has still too large factors to be useful. Fortunately, from time to time, one of them is smooth enough to be factored in the so-called factor base, and will contribute to discovering a factor of n. But this factorization algorithm would become much more efficient if another method was discovered, which finds square roots modulo n of still smaller integers.

Another example is provided by Okamoto and Shiraishi's digital signature scheme [OS]. In this scheme, the signature of the message m is an integer s such that s^2 (mod n) is close to h(m) -where h is a one-way hash-function- instead of being exactly equal, as in the Rabin scheme (the "square root variant" of RSA). The claimed advantage of this scheme was a very fast signature computation compared to the computation time necessary to extract an exact square root modulo n. But Brickell and Delaurentis broke the scheme by showing that s can be efficiently computed, even when the factors of n are hidden [BD]. Now, it remains an open question: can their attack be generalized to the version of Okamoto and Shiraishi's scheme in which the signature s is such that $s^L \# h(m)$ (mod n) with $L \geq 4$?

This paper aims at collecting the results already established concerning these questions and improving them whenever possible. First (section II), we state the problems we are going to deal with. Then (section III), we recall how such problems naturally arose in public-key cryptology and briefly indicate how they were solved (or not...). Finally (section IV), we describe most of the algorithms sketched in section III, generally with enough details to effectively implement them.

II. THE PROBLEMS

What do we mean by approximate L-th roots modulo n? In fact, this includes a lot of various questions amongst which we will consider the following ones (n, L and y_0 are three given positive integers, with $L \geq 2$ and $y_0 < n$):

Firstly, we wish to find an integer whose L-th power modulo n is close to the given integer y_0. We subdivide this problem into three ones:

(1): Find x such that $x^L \# y_0 \pmod n$ (no matter where x stands).
(1a): Given x_0 such that $y_0 = x_0{}^L \pmod n$, find $x \# x_0$ ($x = x_0$) such that $x^L \# y_0 \pmod n$.
(1b): Given x_0, find $x \# x_0$ such that $x^L \# y_0 \pmod n$.

Secondly, we wish to get some information about an (existing but unknown) exact L-th root x_0 of the given integer y_0. We subdivide this problem into both which are complementary :

(2a): Find x such that $x \# x_0$.
(2b): Given x such that $x \# x_0$, find x_0.

(Note that problems (2a) and (2b) cannot be both efficiently solved with the same order of approximation, or there is an efficient algorithm which finds exact L-th roots modulo n.)

Of course, the symbol # may have various significations, upon which depends whether the problem is efficiently solvable or not. In order to be more specific, we state again the above problems by replacing "$x \# x_0$" with "$x = x_0 + O(n^a) \pmod n$" and "$x^L \# y_0 \pmod n$" with "$x^L = y_0 + O(n^b) \pmod n$", where a and b are real numbers picked in the interval]0,1[. Note that, if some of these problems are easily solved when prime factors of n are known, this knowledge apparently does not help to solve other ones, for example (1a) and (1b).

III. HOW PROBLEMS AROSE

III-1. Problem (1) and its variants

As already noted in the introduction, problem (1) was considered by Morrison and Brillhart (using an idea from Lehmers & Powers) with $L=2$ and $y_0=0$ [MB]. By computing continued fractions of $n^{1/2}$, one obtains values of x such that $x^2 (\mathrm{mod}\ n) = O(n^{1/2})$. But only a few of them are useful to factorize n, because the quadratic residues modulo n which are required are generally much smaller than $n^{1/2}$. Unfortunately, no efficient algorithm is known, which solves (1) with $b<1/2$. On the other hand, we show in section IV that continued fraction algorithm can still be used to solve (1) for small exponents greater than 2, but with b growing rapidly with L.

The case $\{L=2;\ \text{any}\ y_0\}$ was solved by Brickell and Delaurentis [BD], when they cryptanalysed Okamoto and Shiraishi's signature scheme [OS]. In this scheme, the signature of the message m (or rather of its hashed version h(m)) is an integer s, not too small in absolute value, and such that $s^2\ (\mathrm{mod}\ n) = h(m) + O(n^{2/3})$. The public modulus n of the signer is in the form $n=p^2q$ (p and q distinct primes), because this permits a very fast computation of s when p and q are known, which was the claimed advantage of this scheme. Unfortunately for it, Brickell and Delaurentis showed that s can be computed with the same efficiency without knowing the factors of n and no matter what form n takes! We will see that their method can be easily extended to any exponent b greater than 1/2 (hence solving the problem (1) for $L=2$, $y_0=0$ and $b>1/2$). Moreover, it solves the variant (1b) with $L=2$ and $b>1/2$, but only if $a+b/2>5/4$. The Brickell and Delaurentis method can also be used for $L=3$ [BO] but does not seem to work for $L\geq 4$.

One year later, the cryptanalysis of some hash-function using modulo-n operations due to Davies and Price is reduced in [G] to solving (1a) for $L=2$ and $a=b=7/8$. First, the problem is linearized by putting $x = x_0 + u$. It can then be stated as follows: find a "very small" u such that $2ux_0 \# 0\ (\mathrm{mod}\ n)$. Now, the equation $2ux_0 = v\ (\mathrm{mod}\ n)$ with small unknowns u and v is solvable by developing $2x_0/n$ in continued fractions, i.e. by applying the extended Euclid's algorithm to the integers $2x_0$ and n. As the solutions provided are shown to be such that $|uv|<n$, the problem (1a) appears to be solved for $L=2$, $a+b\geq 1$ and $b\geq 2/3$.

Does this last method allow us to solve (1b) (and consequently the problem (1) itself) for the same values of L, a and b? In this case, since we have no more $y_0 = x_0^2\ (\mathrm{mod}\ n)$, we are led to an "affine" problem rather than a linear one; explicitly: find a very small u such that $2ux_0 \# z_0\ (\mathrm{mod}\ n)$, where $z_0 = x_0^2 - y_0\ (\mathrm{mod}\ n)$. In [OS], is presented a variant of Euclid's algorithm which allows to do that. Ironically enough, Shamir used it to give another cryptanalysis of the OS signature scheme discussed above and presented in the same paper, as well as the cryptosystem proposed one year after by Okamoto in [O1]!

Recently, the authors have described in [VGT1] (or [VGT2]) a different technique powerful enough to solve also variant (1b) for L=2, a+b>1 and b>2/3. First, the problem is linearized as above. Then, the linear equation (E): $2ux_0 = v \pmod n$ is interpreted as the equation of the integer lattice R spanned by $(1,2x_0)$ and (0,n). This point of view allows us to transform the problems into lattice ones ("find a short vector in R" or "find a point of the lattice close to a given point"), for which algorithmic solutions are known in all dimensions, based on the LLL basis reduction algorithm [LLL]. Moreover, this method can be refined in order to find x in a "quasi-uniform way", leading to a factorisation algorithm [V] whose proven complexity is smaller than Dixon's one [D].

III-2. **Problems (2a) and (2b)**

Problem (2a) with L=2 was first solved by Blum, Blum and Shub [BBS], when they analysed the left-unpredictability of the so-called $x^2 \pmod n$ generator. In their paper, n is a Blum integer so that the mapping "squaring modulo n" is a permutation over the set of quadratic residues modulo n. So, by working in this set, the square root of a quadratic residue is defined in an inambiguous manner. It is shown in [VV] that the location of x_0 (equal to 0 if $x_0<n/2$, 1 if not) cannot be guessed, even with a very small advantage, unless factoring is easy. It follows that (2a) with L=2 is not efficiently solvable for any a<1 since even the location of x_0 cannot be found.

The same problem is solved for all the L which are coprime to phi(n) -the RSA context- by (first) Goldwasser, Micali and Tong [GMT], then many others until Alexi, Chor, Goldreich and Schnorr [ACGS], when they studied the security of the RSA bits. It can be proved that the location of x_0 cannot be guessed, even with a small advantage, unless inverting RSA (i.e. finding x_0 in full) is easy.

The problem (2b) has been partly solved by Shamir when he cryptanalysed the first version of Okamoto's cryptosystem [O1]. In this cryptosystem, the public modulus n is in the form $n=p^2q$, as for Okamoto and Shiraishi's signature scheme. Moreover, the public key contains another integer x, itself of a very particular form. With the notations of (2b), x_0-x plays the role of the plaintext and y_0 is the ciphertext. Shamir found two cryptanalysis for this system. The first one, based on the OS-variant of Euclid's algorithm, does not make use of the form of n but is valid only for L=2. The second one works for any L but does make use of the particular form of n and x. At this stage, the problem (2b) appears to be solved for $\{L=2, a \leq 1/3\}$ and only in very particular cases if $L>2$.

In [VGT1], the authors, using the lattice technique already mentioned in section III.1, specify an efficient algorithm solving (2b) for $\{L=2, a \leq 1/3\}$ and more generally for any reasonably small L, sufficiently large n and sufficiently small a (in the order of about $1/L^2$). So, an L-th root modulo n can always be calculated, if we are given a sufficiently good approximation of this root. Moreover, the technique is general enough to apply to other types of approximations, such as used in the second version of Okamoto's cryptosystem [O2]. This version hence appears to be broken too.

These results have incidences on the predictibility of congruentiel pseudo-random generators. In particular, the truncated x^2 (mod n) generator, obtained by removing the 1/3 least significant bits of the sequence, is right-predictible (in a sense section V will make clear).

IV. THE ALGORITHMS

We now describe the algorithms with more details. Frow now on, n is a positive integer and Z(n) denotes the ring of the integers modulo n that we identify with the interval of length n centered at 0. For any u in Z(n), $|u|$ denotes the absolute value of u, i.e. the maximum of u and -u (for example, $|21 \pmod{25}|=|-4|=4$).

The symbols x, x_0, y, y_0, u denote elements of Z(n) whilst a and b denote real numbers in]0,1[. The notation O(f(n)) stands for any function g(n) such that $|g(n)| \leq kf(n)$ for some integer k and any sufficiently large n.

For L a positive integer, we ask if the equation:

$$x^L = y \bmod n \qquad (E)$$

admit solutions (x,y) which satisfy some closeness requirements, and if we can discover them. For example, can we find (x,y) such that x is "close" to a given integer x_0 and y "close" to another one y_0? Or without conditions on x but such that y is exactly equal to y_0?

Of course, we can (and will often have to...) reduce our ambitions and claim our satisfaction if we succeed in inferring some partial information about such solutions, or, conversely, if some additional information about these solutions permit us to recover them entirely. The most famous case is the computation of an exact square root modulo n (n a large integer), where two extreme situations are possible. Either the factors of n are known and such a computation is child's play; or they are hidden and we can infer almost nothing about the solution.

It must be noticed that most of the algorithms which are presented below do not work for all the values of their inputs. Only in some cases (e.g. algorithms VGT1 and VGT2), the set of values for which they fail has been carefully analysed in the original papers. But the fact they provide solutions *most of the time* is satisfactory enough for cryptologic applications. This is particularly true for cryptanalytic ones, which only require that the algorithm work for a non-negligible fraction of the input values.

IV-1. **Finding roots with small residual**

IV-1.1. **Without conditions for x**

We first consider the problem of finding solutions of (E) where y is close to y_0:

Pb (1): Given n, L, b and y_0, find x such that $|x^L - y_0 \pmod n| \leq O(n^b)$.

Of course, if b is big enough, there is a straight-forward way of finding some of them, which consists in detecting the elements of $[y_0-O(n^b), y_0+O(n^b)]$ which are *true* L-th powers (as opposed to L-th powers *modulo n*) of an integer x. For instance, if L=2 and $b \geq 1/2$, $x=[y_0^{1/2}]$ (where [z] denotes the closest integer to the real number z) is a trivial solution of our problem since:

$$x^2 = (y_0^{1/2} + \beta)^2 \qquad \text{with } |\beta| \leq 1/2$$
$$= y_0 + 2\beta y_0^{1/2} + \beta^2 \quad ==> \quad |x^2 - y_0| \leq y_0^{1/2} + 1 < n^{1/2} \leq n^b.$$

In case b is a little bit too small and $[y_0-O(n^b), y_0+O(n^b)]$ does not contain true L-th powers, other intervals $[y_0+kn-O(n^b), y_0+kn+O(n^b)]$ can be tried, for k=1,2,.... Solutions which are found in this way may be considered as trivial ones. As they are necessarily small, with the same order of magnitude of $n^{1/L}$, we may define trivial solutions of (1) as solutions $x = O(n^{1/L})$. It may occur that the algorithms which are presented below provide trivial solutions, but they (generally) provide also non-trivial ones.

The quadratic version (L=2) of Pb (1) has been solved in the general case by Morrison and Brillhart for $y_0=0$ [MB]. They find approximations in the order of magnitude of $n^{1/2}$. More precisely:

Algorithm MB

Input: n

Output: some x such that $|x^2 \pmod n| \leq 2n^{1/2}$

Method: develop $n^{1/2}$ in continued fractions; call x_i/y_i the convergents of $n^{1/2}$; output the x_i.

Proof: from a well-known inequality of continued fractions theory, we have:

$|n^{1/2} - x_i/y_i| \leq 1/(y_i y_{i+1})$

then:

$|n^{1/2} + x_i/y_i| \leq 2n^{1/2} + 1/(y_i y_{i+1})$

$|n - x_i^2/y_i^2| \leq 2n^{1/2}/(y_i y_{i+1}) + 1/(y_i y_{i+1})^2$

$|y_i^2 n - x_i^2| \leq 2n^{1/2} y_i/y_{i+1} + 1/y_{i+1}^2 \leq 2n^{1/2} \Longrightarrow |x_i^2 \pmod n| \leq 2n^{1/2}$.

Remark: when the periodicity of the development of $n^{1/2}$ is small (for example if $n=m^2+1$), this algorithm only provides trivial solutions; if not, there may be a lot of (non-trivial) solutions.

We now show that the MB idea can be extended to small exponents other than 2, but with less efficiency because only first convergents have a good chance to lead to success :

Algorithm MB'

Input: n, L (small integer = O(log n) and ≥ 3), ß (real $\leq 1/L$)

Output: nothing or some x such that $|x^L \pmod n| \leq L n^{(L-1)/L+\beta}$

Method: develop $n^{1/L}$ in continued fractions; call x_i/y_i the convergents of $n^{1/L}$; output the x_i until (say) $y_i > n^{\beta/(l-2)}$.

Proof: from $|n^{1/L} - x_i/y_i| \leq 1/(y_i y_{i+1})$ we deduce:

$n^{(L-1)/L} + n^{(L-2)/L}(x_i/y_i) + ... + n^{1/L}(x_i/y_i)^{L-2} + (x_i/y_i)^{L-1} \leq Ln^{(L-1)/L} + L^2 n^{(L-2)/L}/y_i y_{i+1}$

then:

$|n - x_i^L/y_i^L| \leq Ln^{(L-1)/L}/(y_i y_{i+1}) + L^2 n^{(L-2)/L}/(y_i y_{i+1})^2$

hence:

$|y_i^L n - x_i^L| \leq Ln^{(L-1)/L} y_i^{(L-1)}/y_{i+1} + L^2 n^{(L-2)/L} y_i^{(L-2)}/y_{i+1}^2 \leq Ln^{(L-1)/L} y_i^{L-2}$

(because the second term of the sum is easily shown to be smaller than $Ln^{(L-1)/L}(y_i^{L-2}-y_i^{L-1}/y_{i+1})$ except perhaps in some very particular cases)
finally:
$|x_i^L \pmod n| \le Ln^{(L-1)/L+\beta}$.

But continued fractions do not seem to help in solving the problem when y_0 is non zero. In [BD], Brickell and Delaurentis describe an algorithm which solves the case $\{L=2;$ any $y_0\}$ with $b=2/3$ (i.e. approximations of $n^{2/3}$). But, as their original algorithm can be easily extended to $b=1/2+e$, for $0<e<1/2$, it appears to be almost as efficient than algorithm MB, and much more general.

The idea is the following one: not only $[y_0^{1/2}]$ is a solution of Pb (1) but also $k[z^{1/2}]$ where $z = y_0k^{-2} \pmod n$, for any positive integer $k \le n^{e/2}$, as shown in the proof below. These new solutions are not trivial ones but are in $O(n^{1/2+e/2})$. In order to find solutions of any magnitude, Brickell and Delaurentis proceed as follows:

Algorithm BD

*Input: n, y_0, e $(0<e<1/2)$
*Output: some x such that $|x^2 - y_0 \pmod n| = O(n^{1/2+e})$

*Step 1: find k (coprime with n) and x' such that $k=O(n^{e/2})$ and $2kx' \pmod n = O(n^e)$;
*Step 2: calculate $y = y_0 - x'^2 \pmod n$
$z = yk^{-2} \pmod n$
$t = [z^{1/2}] = z^{1/2} + \beta$ with $|\beta| \le 1/2$;
*Step 3: output $x = x' + kt$.

*Remark: for step 1, it suffices to choose x' in one of the intervals I_i centered in $[ni/2k]$ of radius $[n^e/2k]$.

*Proof: in a straightforward manner (all the equalities standing modulo n):

$$\begin{aligned}
x^2 &= x'^2 + k^2t^2 + 2kx't \\
&= x'^2 + k^2(z^{1/2}+\beta)^2 + 2kx't \\
&= x'^2 + k^2z + 2k^2z^{1/2}\beta + k^2\beta^2 + 2kx't \\
&= x'^2 + y_0 - x'^2 + 2k^2z^{1/2}\beta + k^2\beta^2 + 2kx't \\
&= y_0 + O(n^{1/2+e}) + O(n^e) + O(n^{1/2+e}) \\
&= y_0 + O(n^{1/2+e})
\end{aligned}$$

Does algorithm BD extend to $L \geq 2$? One can remark that the above proof does not work for L=3, except if one makes some very particular choices, namely : $x' = [n/3]$ and $k = 1$. Then, choosing $y = y_0 - x'^3 \pmod n$ and t = nearest integer to $y^{1/3}$ divisible by 3, leads to success for $b = 2/3$; details may be found in [BO]. For $L \geq 4$, the method does not seem to work at all.

IV-1.2. **With conditions for x**

We now come to algorithms which not only solve Pb (1) but provide solutions which are themselves close to a given integer x_0. This new problem can be subdivided into two subproblems. In first one, Pb (1a), y_0 is nothing but the L-th power of x_0; in other words, we have to find a solution (x,y) of (E), close to another already known solution. In second one, Pb (1b), y_0 and x_0 are any two elements of Z(n). It is clear that this last problem is harder than both Pb (1) and Pb (1a). Note also that knowledge of the factors of n completely solves Pb (1) but does not seem to help to solve Pb (1a) and Pb (1b).

Pb (1a): Given n, L, a, b, x_0 and y_0 such that $y_0 = x_0^L \pmod n$, find $x = x_0$ such that $|x - x_0| \leq O(n^a)$ and $|x^L - y_0 \pmod n| \leq O(n^b)$.

Pb (1b): Given n, L, a, b, x_0 and y_0, find x such that $|x - x_0| \leq O(n^a)$ and $|x^L - y_0 \pmod n| \leq O(n^b)$.

First, a closer look at BD-algorithm shows that it solves (but not very well) Problem (1b):

Algorithm BD'

**Input*: n, x_0, y_0, a, b (s.t. $a+b/2>5/4$ and $b>1/2$)

**Output*: some x such that $|x - x_0| \leq O(n^a)$ and $|x^2 - y_0 \pmod n| \leq O(n^b)$

**Method*: as in algorithm BD with $e=b-1/2$, by choosing k close to $n^{e/2}$ and x' in the interval I_i which is the closest one to x_0.

**Proof*: the distance between two consecutive intervals I_i and I_{i+1} , defined in the remark of algorithm BD, is smaller than $n/2k = O(n^{1-e/2})$.

Better solutions to Pb (1b) are obtained by linearizing it, as will be explained in subsection IV.1.1.2. Beforehand, we have to make a digression into Euclid's algorithm and some of its extensions.

IV-1.2.1. A Euclidean digression

We consider here the equation:

$$dx = y \pmod{n} \qquad (E')$$

where d is a positive integer smaller than n, and ask if there are solutions (x,y) close to a given pair (x_0,y_0): $x = x_0 + O(n^a)$ and $y = y_0 + O(n^b)$.

Let us start with the case $x_0 = y_0 = 0$. It is proven in [DC] (or [G]) that such solutions certainly exist if $a+b = 1$. More precisely, for any pair (X,Y) whose product is greater than n, there is at least one solution (x,y) such that $|x|<X$ and $|y|<Y$. In order to discover it (or them), it is useful to remark that finding small (x,y) satisfying (E') comes to finding a good approximation of the fraction d/n. So, here again, we (almost) always find such a solution by developing it in continued fractions i.e. applying Extended Euclid's algorithm to d and n:

<u>Algorithm EE</u>

**Input*: n, d, a, b (s.t. $a+b\geq1$)

**Ouput*: nothing or some x such that $|x| \leq n^a$ and $|dx \pmod{n}| \leq n^b$

**Method*: apply Extended Euclid's algorithm to n and d; one obtains coefficients l_i and m_i such that $l_i n + m_i d = r_i$ where the r_i are the successive remainders (the last non-zero remainder being equal to the greatest common divisor of n and d); output the smallest (in absolute value) m_i such that $n^{1-b} \leq |m_{i+1}|$ (the case "such a m_i does not exist" is very rare).

**Proof*: the fractions $|l_i/m_i| = -l_i/m_i$ are in fact the convergents of the development of d/n in continued fractions; hence:
$|d/n + l_i/m_i| \leq 1/|m_i m_{i+1}|$ ==> $|dm_i + nl_i| \leq n/|m_{i+1}|$ ==> $|dm_i \pmod{n})| \leq n/|m_{i+1}| \leq n^b$.
Moreover, $|m_i| \leq n^{1-b} \leq n^a$ since $a+b \geq 1$.

Now, what happens if x_0 and y_0 are non-zero? Of course, it is enough to solve the problem for $x_0 = 0$ and any y_0 (if x_0 is not equal to zero, it suffices to replace y_0 with y_0-$dx_0 \pmod{n}$). Okamoto and Shiraishi provide in [OS] an extension of Extended Euclid's algorithm which very often solves this problem. We hope that we do not deform it too much by presenting it as follows:

Algorithm OS

*$Input$: n, d, a, b (s.t. $a+b\geq 1$), y_0
*$Ouput$: nothing or some x such that $|x| \leq n^a$ and $|dx - y_0 \pmod n| \leq n^b$

*$Step\ 1$: apply Extended Euclid's algorithm to d and n (as in algorithm DC);
*$Step\ 2$: introduce a sequence y whose first term is y_0 and following ones are defined by: $y_i = y_{i-1} - q'_i r_i$ where q'_i is the quotient in the division of y_{i-1} by r_i;
*$Step\ 3$: introduce also the sequences h_i and k_i whose first terms h_0 and k_0 are zero and following ones are defined by: $h_i = h_{i-1} + q'_i l_i$ and $k_i = k_{i-1} + q'_i m_i$;
*$Step\ 4$: output k_i such that $n^{1-b} \leq |k_i| \leq n^a$ (mind: its existence is questionable, especially if a+b is close to 1).

**Proof*: From: $l_i n + m_i d = r_i$, we easily deduce:
$h_i n + k_i d = h_{i-1} n + k_{i-1} d + (y_{i-1} - y_i)$; then :
$h_i n + k_i d = 0 + (y_0 - y_1) + (y_1 - y_2) + ... + (y_{i-1} - y_i) = y_0 - y_i \Longrightarrow k_i d \pmod n = y_0 - y_i$.
Moreover, when it can be shown that $|k_i| y_i < n$ (it is very often the case), we have : $|k_i| \geq n^{1-b} \Longrightarrow y_i \leq n^b$.

IV-1.2.2. **Come back to our problems**

In [G], one of the authors shows that the quadratic version of (1a) can be solved with $a+b\geq 1$ and $b\geq 2a$ (which is equivalent to: $a+b\geq 1$ and $a\leq 1/3$ or: $a+b\geq 1$ and $b\geq 2/3$). The idea consists in reducing the problem to a linear one by taking advantage of the fact we already know a solution of (E):

Algorithm G

**Input*: n, a, b (s.t. $a+b\geq 1$ and $b\geq 2a$), x_0, y_0 (s.t. $y_0 = x_0^2 \pmod n$).
**Output*: nothing or some x such that $0 < |x - x_0| \leq n^a$ and $|x^2 - y_0 \pmod n| \leq 2n^b$)

**Method*: perform algorithm EE with inputs n, $2x_0$, a, b; output $x = x_0 + m_i$.

**Proof*: $x^2 = (x_0 + m_i)^2 = x_0^2 + m_i^2 + 2m_i x_0^2 \quad \pmod n$
$\qquad = y_0 + m_i^2 + 2m_i x_0 \quad \pmod n$
We know from previous section that: $|2m_i x_0 \pmod n| \leq n^b$.
Moreover, $|m_i| \leq n^a \Longrightarrow m_i^2 \leq n^{2a} \leq n^b$ since $b\geq 2a$.
It follows that $|x^2 - y_0 \pmod n| \leq 2n^b$.

Let us now consider Pb (1b). If, in algorithm G, we substitute algorithm OS to Euclid's one, we obtain a new algorithm that Shamir used to cryptanalyse OS signature scheme:

Algorithm S1

**Input*: n, a, b (s.t. $a+b\geq 1$ and $b\geq 2a$), x_0, y_0

**Output*: nothing or some x such that $|x - x_0| \leq n^a$ and $|x^2 - y_0 \pmod n| \leq 2n^b$

**Method*: perform algorithm OS with inputs n, $2x_0$, a, b, $z_0 = y_0 - x_0^2 \pmod n$; output $x = x_0 + k_i$.

**Proof*: $x^2 = (x_0 + k_i)^2 = x_0^2 + k_i^2 + 2k_ix_0 \pmod n$

We know from previous section that, very often: $|2k_ix_0 - z_0 \pmod n| \leq n^b$.

Moreover, $|k_i| \leq n^a \Longrightarrow k_i^2 \leq n^{2a} \leq n^b$, since $b\geq 2a$.

It follows that $|x^2 - y_0 \pmod n| \leq 2n^b$.

Another point of view has been recently considered by the authors in [VGT1 or 2]. Using the theory of lattice basis reduction, they present an algorithm which solves Pb (1b) in the same conditions as algorithm S1 but is more adapted to generalizations (see IV.2). The starting point is identical: we want $x = x_0 + u$, and $x^2 \pmod n = y_0 + v$ with u and v small. These two equalities imply: $2ux_0 = z_0 + w \pmod n$ with $z_0 = y_0 - x_0^2 \pmod n$ and $w = v - u^2 \pmod n$. But the set of vectors (u,u') such that $2ux_0 = u' \pmod n$ may be seen as the lattice $R(x_0)$ spanned by vectors $(1,2x_0)$ and $(0,n)$. If we find a point (u,u') of $R(x_0)$ close to $(0,z_0)$ in that $u = O(n^a)$ and $u' = z_0 + O(n^b)$, then $v = w+u^2 = O(n^b) + O(n^{2a}) = O(n^b)$, since $b\geq 2a$, and the problem is solved. We now see how to find such a point (u,u'):

Algorithm VGT1

**Input*: n, a, b (s.t. $a+b>1$ and $b\geq 2a$), x_0, y_0

**Output*: nothing or some x such that $|x - x_0| \leq O(n^a)$ and $|x^2 - y_0 \pmod n| \leq O(n^b)$

**Step 1*: consider the lattice $M(x_0)$ spanned by vectors $(k,2k'x_0)$ and $(0,k'n)$ where $k = [n^{(b-a)/2}]$ and $k' = 1/k$.

**Step 2*: use LLL algorithm [LLL] to find a point (t,t') of $M(x_0)$ very close to the point $(0,k'z_0)$ with $z_0 = y_0 - x_0^2 \pmod n$.

**Step 3*: output $x = x_0 + t/k$ (mind: its existence is questionable, especially if a+b is close to 1).

**Proof*: let $e = (a+b-1)/2$. Except in some exceptional cases (see details in [VGT1]), the shortest vector of the lattice $M(x_0)$ is not too small, of length $\geq n^{1/2-e}$. The lattice theory tells us that any ball $B(P,r)$ with $r > (2H_2)^{1/2}n^{1/2+e}$, where H_2 is Hermite's constant in dimension 2 (for definition see e.g. [LLL] or [VGT2]), contains at least one point of the lattice. Moreover, LLL algorithm (i.e. Gauss' algorithm in dimension 2) allows us to find such a point, say $T(t,t')$. Let P be the point $(0,k'z_0)$. Since the distance between T and P is smaller than r, we have $|t|$ and $|t'-k'z_0| \leq (2H_2)^{1/2}n^{1/2+e}$. It remains to put $u=t/k$ and $u'=t'/k'$ (remark that u and u' are necessarily integers). Then: $|u| \leq (2H_2)^{1/2}n^{1/2+e}n^{(a-b)/2} \leq O(n^a)$ and $|u'-z_0| \leq (2H_2)^{1/2}n^{1/2+e}n^{(b-a)/2} \leq O(n^b)$, the inequalities we wanted.

IV-2. Finding something about exact roots

We now consider the problem of finding x_0, an exact L-th root modulo n of a given integer y_0. Here again, there are situations in which the problem can be considered as trivial: when y_0 is a *true* L-th power or when factorization of n is known. At the opposite, the problem is specially hard in almost all other cases, since one (presently) does not know how extract L-th roots modulo n without factors of n. Between these two extreme situations, we may consider intermediary ones. First, can we at least infer some partial information about where stands x_0? Or, on contrary, if we are given some information about location of x_0, can we recover it entirely?

IV-2.1. Inferring some partial information about location of x_0

We first consider the following problem:

Pb (2a): Given n, L, a and y_0 (known to be the L-th power modulo n of an integer x_0), find x such that $|x - x_0| \leq O(n^a)$.

This question has been widely discussed between 1982 and 1984, since it is related to security of RSA (or Rabin) bits [BBM],[GMT],[VV],[ACGS]. The conclusion was that Pb (2a) has definitely no solution at all, even when a is very close to 1 (we refer the reader to the introduction).

IV-2.2. Finding x_0 with some help

Let us come now to our last problem:

Pb (2b): Given n, L, a, y_0 (known to be the L-th power modulo n of an integer x_0), and x

Pb (2b): Given n, L, a, y_0 (known to be the L-th power modulo n of an integer x_0), and x such that $|x - x_0| \leq O(n^a)$, find x_0.

In [O2], Okamoto presents an algorithm due to Shamir which solves the quadratic version of this problem (it is the little sister of algorithm S1):

Algorithm S2

**Input*: n, y_0, x (s.t. $|x - x_0| \leq O(n^{1/3})$)

**Output*: nothing or x_0

**Method*: apply algorithm OS to n, $2x_0$, 1/3, 2/3, $z_0 = x^2 - y_0 \pmod n$; output $x = x_0 + k_i$ for $k_i \# n^{1/3}$; check that $y_0 = x_0^2 \pmod n$.

**Proof*: let $x = x_0 + u$; then $2x_0u = z_0 - u^2 \pmod n$ and we are reduced to finding $u = O(n^{1/3})$ such that $|2x_0u - z_0 \pmod n| \leq O(n^{2/3})$. Such an u is likely to be one of the k_i close to $n^{1/3}$, provided by algorithm OS.

In [VGT1], the authors, using the lattice technique inroduced in IV-1. solve Pb (2b) for any (reasonably small) L and a in the order of about $2/L^2$. We only state here a weak version of this result (more generally, x_0 can be found even if y_0 is not exactly known, provided the approximation on y_0 is in the order of about 2/L) and we suppose n square-free for simplicity:

Algorithm VGT2

**Input*: n (square-free), $a=2/[L(L+1)]$, y_0, x (s.t. $|x - x_0| \leq O(n^a)$)

**Output*: nothing or x_0

**Sketch of the method*: similar to algorithm [VGT1] (but mind: now x is known and x_0 is the unknown!).

**Sketch of the proof* : similar to VGT1; the property used here is the following one: if the shortest vector of M(x) is not too small, of length $\geq n^{1/L-\epsilon}$, then the ball B(P,r) with $r < n^{(1/L-\epsilon)}/2$ contains at most one point of the lattice. This point is found, if it exists and lies in a slightly smaller ball, by LLL algorithm.

To be more explicit, let us consider the case L=3. In that case, the lattice R(x) is the one spanned by the vectors $(1,0,3x^2)$, $(0,1,3x)$ and $(0,0,n)$ and the lattice M(x) is obtained by multiplying the first (resp. second, resp. third) coordinate by k (resp. k', resp. k") such that $kn^a = k'n^{2a} = k''n^b$ and $kk'k''=1$.

The general result of [VGT1] can interestingly be applied to pseudo-random number generators (PRNG). Consider for example the case L=2 (hence a=1/3) and the sequence : $s_{i+1} = s_i^2$ (mod n) where s_0 is a secret seed. Let t_i be the number obtained by removing the $[(\log_2 n)/3]$ least significant bits of s_i, for $i \geq 1$. It results from [VGT1] that this PRNG is not secure, since we can recover s_1 and s_2 (hence all the s_i, hence all the t_i!) from t_1 and t_2.

V. CONCLUSION

We have shown how various known algorithms may be considered as variants of the computation of approximate L-th roots modulo n, and have given a unified description of all these (often revisited) algorithms. Except for the most complicated ones, for which given references should be consulted, enough details are provided to implement them.

On the way, we have improved or extended some of these algorithms (see algorithm MB', extension of algorithm BD to any small exponent e, algorithm BD', and "new look" of algorithm OS).

Some questions remain open amongst which we point out:

1) can we solve $x^L \# y_0$ (mod n) for $L \geq 4$ with an approximation on y of order $n^{2/3}$ (this problem is related to generalized Okamoto-Shiraishi signature scheme) ?

2) can we solve $x^2 \# y_0$ (mod n) with an approximation on y of order less than $n^{1/2}$ (this problem is related to Morrison-Brillhart factorization algorithm) ?

3) can we solve $x \# x_0$ and $x^2 \# y_0$ (mod n) with an approximation on x and y of order $n^{1/2}$ (this problem is related to quadratic congruentiel pseudo-random number generator) ?

VI. ACKNOWLEDGEMENTS

We thank one member of the program committee for having inspired the redaction of our abstract.

VII. REFERENCES

[ACBG] W. Alexi, B. Chor, O. Goldreich and C.P. Schnorr, "RSA and Rabin functions: certain parts are as hard as the whole", SIAM J. Comp., Vol. 17, pp. 194-209, 1988.

[BBS] L. Blum, M. Blum and M. Shub, "Comparison of two pseudo-random number generators", Advances in Cryptology, Proc. of Crypto 82, Plenum press, New York, 1983, pp.61-78.

[BO] E. Brickell and A. Odlyzko, "Cryptanalysis: a survey of recent results", Proc. of the IEEE, Vol. 76, n°5, May 1988, pp. 578-593.

[BD] E. Brickell and J. Delaurentis, "An attack on a signature scheme proposed by Okamoto and Shiraishi", Proc. of Crypto '85, LNCS, Vol. 218, Springer-Verlag, 1986, pp.10-14.

[D] J.D. Dixon, "Asymptotical fast factorization of integers", Math. Comp., Vol. 36, 1981, pp. 255-260.

[DC] W. De Jonge and D. Chaum, "Attacks on some RSA signatures", Advances in Cryptology, Proc. of Crypto '85, LNCS, Vol. 218, Springer-Verlag, 1986, pp.18-27.

[G] M. Girault, "Hash-functions using modulo-n operations", Proc. of Eurocrypt '87, LNCS, Vol. 304, Springer-Verlag, 1988, pp. 217-226.

[GMT] S. Goldwasser, S. Micali and P. Tong, "Why and how to establish a private code on a public network", Proc. of the 23rd IEEE FOCS, 1982, pp. 134-144.

[LLL] A.K. Lenstra, H.W. Lenstra and L. Lovasz, "Factoring polynomials with integer coefficients", Mathematische Annalen, 1982, Vol. 261, pp. 513-534.

[MB] M.A. Morrison and J. Brillhart, "A method of factorization and the factorization of F_7", Math. Comput., Vol. 29, 1975, pp. 183-205.

[O1] T. Okamoto, "Fast public-key cryptosystem using congruent polynomial equations", Electronics Letters, 1986, Vol.22, pp. 581-582.

[O2] T. Okamoto, "Modification of a public-key cryptosystem", Electronics Letters, 1987, Vol. 23, pp.814-815.

[OS] T. Okamoto and A. Shiraishi, "A fast signature scheme based on quadratic inequalities", Proc. of the 1985 Symposium on Security and Privacy, Apr.1985, Oakland, CA.

[RSA] R.L. Rivest, A. Shamir and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems", CACM, Vol. 21, n°2, Feb. 1978, pp. 120-126.

[V] B. Vallée "Integer factorisation with quasi-uniform generation of small quadratic residues", presented at "Computational Number Theory" Conference, Bodwin College, Jul. 88; submitted to Compte Rendus de l'Académie des Sciences de Paris (preprint available from the author).

[VGT1] B. Vallée, M. Girault and P. Toffin, "How to break Okamoto's cryptosystem by reducing lattice bases", Proc. of Eurocrypt '88, to appear.

[VGT2] B. Vallée, M. Girault and P. Toffin, "How to guess L-th roots modulo n by reducing lattice bases", Proc. of Conference of ISSAC-88 and AAECC-6, Jul. 88, to appear.

[VV] U.V. Vazirani and V.V. Vazirani, "Efficient and secure pseudo-random number generation", Advances in Cryptology, Proc. of Crypto '84, LNCS, Vol. 196, Springer-Verlag, 1985, pp.193-202.