Int. J. Communications, Network and System Sciences, **2011, 4, 133-138**
doi:10.4236/ijcns.2011.43016 Published Online March 2011 (http://www.SciRP.org/journal/ijcns)

Information Protection Based on Extraction of Square Roots of Gaussian Integers

Boris S. Verkhovsky
Computer Science Department, New Jersey Institute of Technology, University Heights, Newark, USA
E-mail: *verb73@gmail.com*

Received February 13, 2011; *revised February* 20, 2011; *accepted February* 22, 2011

Abstract

A cryptosystem, based on computation of square roots of complex integers modulo composite *n*, is described in this paper. This paper provides an algorithm extracting a square root of Gaussian integer. Various properties of square roots and a method for finding Gaussian generators are demonstrated. The generators can be instrumental in constructing other cryptosystems. It is shown how to significantly reduce average complexity of decryption per each block of ciphertext.

Keywords: Public Key Cryptosystems, Square-Root Extraction, Gaussian Integers, Gaussian Generator, Multiplicative Inverse, Square Root Algorithm, Information Hiding, Ambiguity of Recovery

1. Introduction

In recent publications public-key cryptography (PKC) algorithms are generalized in the field of complex integers that were introduced and analyzed by Carl F. Gauss [1]. In this paper complex numbers are denoted as

$$(a,\ b) := a + bi.$$

Definition 1: If all components in $(a,\ b)$ are integers, then such complex number is called a Gaussian integer [1].

Applications of Gaussian integers {Gaussians, for short} in PKC were extended to the RSA scheme [2], and to ElGamal cryptosystems [3]. While groups based on real integers have cycles of order $\mathrm{O}(p)$, groups based on Gaussians modulo prime *p* have cycles of order $\mathrm{O}(p^2)$. In this paper is described a cryptosystem based on extractors of square roots of complex integers modulo composite *n*. The kernel of the proposed application is a method for extracting square roots of a Gaussian modulo n_k, where $n_k = p_k q_k$, p_k and q_k are distinct and large primes.

2. Gaussian Integers

2.1. Arithmetic of Gaussian Integers

Modulo reduction:

$$(a,\ b) \bmod n \equiv (a \bmod n, b \bmod n);$$

Multiplication: Let

$$(g,\ h) := (a,b)\times(c,d) \bmod n;\ A := ac;\ B := bd;$$

$C := a + c$; $D := b + d$; then $g := (A - B) \bmod n$; and $h := (CD - A - B) \bmod n$ [4].

Modular multiplicative inverse:

If $\gcd(a^2 + b^2, n) = 1$, then there exists (*r*, *s*) such that $(a,b)(r,s) \bmod n = (1,0)$.

Gaussian (*r*, *s*) is called a modular multiplicative inverse of (*a*, *b*) [5]; and

$$(r,\ s) := (a,\ p - b)(a^2 + b^2)^{-1} (\bmod\ n). \qquad (2.1)$$

2.2. Square Root of Gaussians

(*x*, *y*) is called a square root of (*c*, *d*) modulo *p* if

$$(x, y)^2 \bmod p = (c, d). \qquad (2.2)$$

Let in (2.2) Gaussian (*c*, *d*) be an *input* and Gaussian (*x*, *y*) be an unknown *output*. From multiplication of two complex numbers follows that

$$(x,\ y)^2 \equiv (x^2 - y^2,\ 2xy) \bmod p. \qquad (2.3)$$

Hence (2.2) and (2.3) imply that *x* and *y* satisfy

$$(x^2 - y^2) \bmod p = c; \qquad (2.4)$$

and

$$2xy \bmod p = d. \qquad (2.5)$$

As in the case of real integers, not every Gaussian modulo *p* has a square root.

Definition 2: If a square root of (*c*, *d*) exists, then (*c*, *d*) is called a *Gaussian quadratic residue* (*GQR*); otherwise (*c*, *d*) is called a *Gaussian quadratic non-residue* (*GQNR*).

2.3. Gaussian Integers Modulo Non-Blum Prime *p*

If a prime *p* satisfies *p*mod4 = 3, then it is called a Blum prime. Using Euler criterion, it is easy to verify that, if *p*mod4 = 1, then $p-1$ is a *QR* modulo *p* [6]. Therefore $\sqrt{-1} \equiv \sqrt{p-1} \pmod{p}$ is a *real* integer.

***Corollary* 1:** All Gaussians modulo a non-Blum prime are *real* integers.

In this paper, only Blum prime moduli are considered.

***Example* 1**: If *p* = {41, 53, 61}, then the corresponding $i \equiv \sqrt{-1} \pmod{p}$ are equal ± {9, 23, 11}. *Corollary*1 implies that for every *c* and *d* $(c+id) \bmod 61 = c \pm 11d$, *i.e.*, (*c*, *d*) is a *real* integer.

2.4. Properties of Square Roots

Proposition 1: Let $(a,\ b)^2 \bmod p = (c,\ d)$ (2.2). Then the following properties hold:

1) $(p-a, b)^2 \bmod p = (c,\ p-d)$; and

$$(a,\ p-b)^2 \bmod p = (c,\ p-d); \tag{2.6}$$

2) If $(w,0)^2 \bmod p = (c,0)$, then

$$(0,\ w)^2 \bmod p = (p-c,\ 0); \tag{2.7}$$

3) If *p*mod4 = 3, then an integer *c* is either a quadratic residue (*QR*) with *two* real roots or a quadratic non-residue (*QNR*). In the latter case (*c*, 0) is a *GQR* with two *imaginary* roots. In other words, if *c* is a *QR*, then *p-c* is a *GQR* and vice verse: if $\sqrt{c} \bmod p = \pm a$, then

$$\sqrt{p-c} \bmod p = \pm(0,\ p-a); \tag{2.8}$$

4) For every Blum prime *p* there exist exactly $(p^2-1)/2$ *GQRs* and $(p^2-1)/2$ *GQNRs*;

5) If $\sqrt{(c,d)} \bmod p = \pm(a,b)$, then

$$\sqrt{(p-c,\ d)} \bmod p = \pm(b,\ a) \tag{2.9}$$

and

$$\sqrt{(c,\ p-d)} \bmod p = \pm(p-a,\ b); \tag{2.10}$$

6) A *GQR* (0, *d*) has two roots $(x,\ \pm y)$ such that

$$x^2 \equiv y^2 \pmod{p}. \tag{2.11}$$

These properties yield from the identity

$$(a,\ b)^2 \equiv (a^2-b^2,\ 2ab) \pmod{p} \tag{2.12}$$

and from Euler criterion of quadratic residuosity [6].

***Remark* 1:** As it is demonstrated in Section 5, the 6th property can be applied in search for a Gaussian generator.

2.5. Symbolic Rules

Many properties including (2.6)-(2.11) are easier to verify if we define a generalization of Legendre symbol [7] for Gaussian integers.

Definition 3:

$$\left(\frac{(c,d)}{n}\right) = \begin{cases} 1, \text{ if } (c,\ d) \text{ is GQR } \bmod n \\ -1, \text{ if } (c,\ d) \text{ is GQNR } \bmod n. \end{cases} \tag{2.13}$$

Proposition 2: If *p*mod4 = 3, and *q*mod4 = 3, then the following properties hold:

1) $$\left(\frac{(c,d)(e,f)}{p}\right) = \left(\frac{(c,d)}{p}\right)\left(\frac{(e,f)}{p}\right); \tag{2.14}$$

2) $$\left(\frac{(c,0)}{p}\right) = \left(\frac{(0,d)}{p}\right) = 1 \quad [8]; \tag{2.15}$$

3) $$\left(\frac{(c,d)}{pq}\right) = \left(\frac{(c,d)}{p}\right)\left(\frac{(c,d)}{q}\right). \tag{2.16}$$

3. Extraction of Square Roots

3.1. Algorithm

The following algorithm describes *how* to extract Gaussian square roots of (*c*, *d*) if it is a GQR modulo Blum prime *p*.

1) Compute

$$L := (p+1)/4; \tag{3.1}$$

if *c* > 0 **and** *d* = 0, **then** assign

$$E := c^L \bmod p; \tag{3.2}$$

2) **if** $E^2 \bmod p = c$, **then** $(x,y) := \pm(E,\ 0)$ **else**

$$(x,\ y) := \pm\left(0,\ (-1)^L E\right); \tag{3.3}$$

3) **if** *c* = 0, *d* > 0, **then** assign

$$E := \left[d^{-1}(p+1)/2\right]^L \bmod p; \tag{3.4}$$

4) **if** $E^2 \equiv d^{-1}(p+1)/2 \pmod{p}$, **then**

$$(x,\ y) := \pm(E,\ E) \tag{3.5}$$

else

$$(x,\ y) := \pm(p-E,\ E); \tag{3.6}$$

5) **if** $c > 0$ and $d > 0$, **then**

$$N := \left(c^2 + d^2\right) \bmod p; \tag{3.7}$$

$$\left\{N \text{ is a norm of } (c,\ d); \left\|(c,\ d)\right\| \equiv N\right\}$$

6) assign

$$A := N^L \bmod p; \tag{3.8}$$

if

$$A^2 \bmod p \neq N, \tag{3.9}$$

then (c, d) does *not* have a square root; {**end** of algorithm}; **else** $g := \left[(A + c)(p + 1)/2\right] \bmod p$;

$$E := g^L \bmod p; \tag{3.10}$$

7) **if** $E^2 \bmod p = g$, **then** $x := E \bmod p$;

$$y := Ed^{-1}(A - c) \bmod p; \tag{3.11}$$

8) **if** $E^2 \bmod p = p - g$, **then**

$$x := Ed^{-1}(c - A) \bmod p;\ y := E \bmod p; \tag{3.12}$$

Output *two* solutions $\pm(x, y)$; {**end** of algorithm}.

***Remark* 2:** System of Equations (2.4)-(2.5) has a closed-form solution: if A is pre-computed in (3.8), then

$$x \equiv \pm\sqrt{(c \pm A)(p + 1)/2} \pmod{p}; \tag{3.13}$$

$$y \equiv \pm\sqrt{(-c \pm A)(p + 1)/2} \pmod{p}, \tag{3.14}$$

where computation of x and y in (3.13)-(3.14) requires *three* exponentiations (3.20). In addition, these square roots in (3.8), (3.13) and (3.14) have four ($\pm$) signs. Hence, there are *sixteen* combinations of these signs. However, not all combinations of these signs are feasible. Feasibility of these combinations is analyzed in Subsection 3.3.

3.2. Algorithm Validation

In the following discussion it is assumed that both components in (c, d) are strictly positive integers. Since the left sides of both Equations (2.4) and (2.5) are homogeneous, consider

$$y := xu; \tag{3.15}$$

where a positive integer u is unknown.

Then Equations (2.4) and (2.5) imply that

$$x^2\left(1 - u^2\right) \bmod p = c; \tag{3.16}$$

and

$$2x^2 u \bmod p = d. \tag{3.17}$$

Then (3.16) and (3.17) imply that

$$d\left(1 - u^2\right) \bmod p = 2cu. \tag{3.18}$$

Hence, $$u := d^{-1}(\pm A - c) \bmod p, \tag{3.19}$$

where

$$A := \sqrt{c^2 + d^2} \equiv \left(c^2 + d^2\right)^{(p+1)/4}. \tag{3.20}$$

Therefore, (3.18)-(3.19) imply that

$$u^{-1} := d^{-1}(\pm A + c) \bmod p. \tag{3.21}$$

Indeed, $u^{-1}u \equiv d^{-2}\left(A^2 - c^2\right) \equiv 1 \pmod{p}$.

Then $$x^2 = \left[(\pm A + c)(p + 1)/2\right] \bmod p, \tag{3.22}$$

which finally yields (3.13), and (3.15) implies (3.14). In addition, algorithm (3.1)-(3.12) requires only *two* extractions of the square roots.

***Example* 2**: Let $(x, y) := \sqrt{(6,1)} \bmod 11$.

Compute $N = 4$, $L = 3$ and $A = 9$ (3.8).

Since $A^2 \bmod p = 4 = N$, then (6,1) is GQR.

Compute $g = 2$, $E := g^L \bmod p = 2^3 = 8$.

Since $E^2 \bmod p = p - g = 9$, then $y := 8$ and $x := Ed^{-1}(c - A) \bmod 11 = 9$. Another solution is $-(9,8) \bmod 11 = (2,3)$.

3.3. Feasibility Analysis

As noticed in *Remark*3, there are $2^4 = 16$ combinations of signs ($\pm$) in (3.13)-(3.14). In order to determine which of these combinations are feasible, rewrite Equations (3.13)-(3.14) in generic form:

$$v = q\sqrt{(rA + c)(p + 1)/2} \bmod p; \tag{3.23}$$

and

$$z = s\sqrt{(tA - c)(p + 1)/2} \bmod p. \tag{3.24}$$

Then variables v and z satisfy (2.4)-(2.5) if and only if variables q, r, s and t satisfy equations:

$$qs \equiv rt \equiv q^2 \equiv s^2 \equiv 1 \pmod{p}. \tag{3.25}$$

As a result, there are only *four* pairs of variables that *may* satisfy (2.4) and (2.5):

$$u_1 \equiv u_2 \equiv (A - c)d^{-1} \pmod{p}$$

and $$u_3 \equiv u_4 \equiv p - (A + c)d^{-1} \pmod{p}. \tag{3.26}$$

Then **either** $x_{1,2} \equiv \pm\sqrt{(A + c)2^{-1}} \pmod{p}$ **and**

$$y_{1,2} \equiv x_{1,2}u_{1,2} \equiv \pm\sqrt{(A - c)2^{-1}} \pmod{p}; \tag{3.27}$$

or $x_{3,4} \equiv \pm\sqrt{(-A + c)2^{-1}} \pmod{p}$ **and**

$$y_{3,4} \equiv x_{3,4}u_{3,4} \equiv \pm\sqrt{(-A - c)2^{-1}} \pmod{p}. \tag{3.28}$$

Thus, if (c, d) is a *GQR*, only *two* of these *four* pairs are solutions of (2.4) and (2.5). An alternative extractor of square roots is provided in [8].

Proposition 3: Let p be a Blum prime and h be a positive integer co-prime with p. Then from Euler criterion of quadratic residuosity [6] either h or $p-h$, but not both, is a *QR* modulo *p*. Indeed, since $(p-1)/2$ is an *odd* integer, then

$$h^{(p-1)/2} \equiv -(p-h)^{(p-1)/2} \pmod p. \qquad (3.29)$$

Proposition 4: Let $A^2 = N$, (3.8), and let $z := c$ or $z := d$; **if** $(A+z)$ is a *QR*, **then** $(A-z)$ is also a *QR*; **else if** $(A+z)$ is a *QNR*, **then** $(A-z)$ is also *QNR*. Indeed, from Euler criterion [6]

$$(A+z)^{\frac{p-1}{2}}(A-z)^{\frac{p-1}{2}} \equiv (A^2-z^2)^{\frac{p-1}{2}} \equiv \begin{cases} (d^2)^{\frac{p-1}{2}} \equiv d^{p-1} \equiv 1 \pmod p \text{ if } z=c \\ (c^2)^{\frac{p-1}{2}} \equiv c^{p-1} \equiv 1 \pmod p \text{ if } z=d \end{cases} \qquad (3.30)$$

Thus $(A+z)^{(p-1)/2} \equiv (A-z)^{(p-1)/2} \pmod p$ (3.31)

***Corollary* 2: If**

$$\left[(A+c)2^{-1}\right]^{(p-1)/2} \bmod p = 1, \qquad (3.32)$$

then (x_1, y_1) and (x_2, y_2) are square roots **else** (x_3, y_3) and (x_4, y_4) are square roots of (*c*, *d*).

Proposition 5: A square root of (*c*, *d*) modulo *p* exists if and only if there exists a square root of norm *N* of (*c*, *d*) [9] and algorithm (3.1)-(3.12) is a constructive procedure for computing it.

4. PKC Based on Square Roots

4.1. System Design

Step 1: A pair of large and distinct Blum primes *p* and *q* are independently selected by each user.

Step 2: *n*: = *pq*; *n* is a public key; *p* and *q* are private keys;

Step 3: Every user pre-computes $M := p^{-1} \bmod q$; and $W := q^{-1} \bmod p$.

4.2. Encryption

Step 4: A sender (called *Alice*) divides plaintext into blocks and represents each block in numeric form as a positive integer *m* < *n*; pairs of integers $m_1, m_2; \cdots; m_{2i-1}, m_{2i}; \cdots$ are treated as Gaussians $(a_i, b_i) := (m_{2i-1}, m_{2i}); \cdots i = 1, 2, \cdots$

Step 5: Suppose Alice wants to send an array $\{(a_1, b_1); (a_2, b_2); \cdots; (a_k, b_k)\}\cdots$ of Gaussians to a receiver (called *Bob*): *Alice* computes *ciphertexts*

$$(c_1, d_1) := (a_1, b_1)^2 \bmod n; \qquad (4.1)$$

$$(c_i, d_i) := (a_1, b_1)(a_i, b_i) \bmod n; \qquad (4.2)$$

and transmits (c_1, d_1) and $(c_{2\le i\le k}, d_{2\le i\le k})$ over open channels to *Bob*.

4.3. Decryption

Step 6: Using the algorithm (3.1)-(3.12) *Bob* finds square roots (u_1, w_1) of (c_1, d_1) modulo *p* and then modulo *q*;

Step 7: *Bob* finds square root (u_1, w_1) of (c_1, d_1) modulo composite *n* by using the Chinese Remainder Theorem (CRT) and pre-computed *M* and *W* in **Step 3**, [1];

***Remark* 3:** An efficient implementation of CRT is provided in [9];

Step 8: *Bob* computes a multiplicative inverse of (u_1, w_1) [5]:

$$(u_1, w_1)^{-1} \equiv (u_1, n-w_1)\left(u_1^2 + w_1^2\right)^{-1} \pmod n; \qquad (4.3)$$

Step 9:

$$(a_i, b_i) \equiv (c_i, d_i)(u_1, w_1)^{-1} \pmod n. \qquad (4.4)$$

Step 10:

$$(a_1, b_1) = (u_1, w_1); \qquad (4.5)$$

***Remark* 4:** In (4.2) (a_1, b_1) is used to hide the plaintexts $(a_{2\le i\le k}, b_{2\le i\le k})$ from an intruder. Since the computation of multiplicative inverse (4.3) is computationally simpler than extraction of square root of (c, d), therefore this way of information hiding significantly reduces average complexity of decryption per block of the ciphertext.

4.4. Advantages of PKC Based on Gaussian Integers

Currently a "tail" of 64 bits is added to every block of a plaintext *P* in order to resolve ambiguity of Rabin decryption [10]. In the PKC based on square-roots of Gaussian integers for the same level of assurance we need to add totally 64 bits to every *Gaussian* integer (rather than to every real integer as in [10]). The cryptosystem proposed in this paper requires $P/(n-32)$ plaintext blocks while the PKC in [10] requires $P/(n-64)$ blocks.

5. Ambiguity in Recovery of Original Information and Its Elimination

A method of "tails" introduced in [10] does not always work. Consider a plaintext (*a*, *b*) = (275, 346) and *p* = 6221. Using tails by repeating the last rightmost digits, we re-write

$$(A, B) := 10(a,b) + (a,b) \bmod 10 = (2755, 3466).$$

The direct computation shows that

$$(2755, 3466)^2 = (3466, 2755)^2 \pmod{6221}$$

Hence, on the decryption stage, if Bob gets (2755, 3466) and (3466, 2755), there is no way to decide which of two Gaussians is authentic. In general, for p = 6221 there are several hundreds Gaussians that have this property, which creates ambiguity. Indeed, consider a Gaussian with three-digit components (*uvw*, *xyz*), where *u*, *v*, ···, *z* are their decimal digits; let $uv + xy = 61$ and $w + z = 11$. Then with one-digit tails we have (uv$\underline{w}$**w**, xy$\underline{z}$**z**). There are 480 Gaussians that have the property described above.

The way to resolve the ambiguity is to label the Gaussians *asymmetrically*: we use prefix of the first component to create a tail [11]. It seems that this approach does not completely eliminate ambiguity. For instance, if p = 6221 and (a,b) = (474,147), then (A,B) = ($\underline{4}$74**4**, 14$\underline{7}$**7**). Yet, it is obvious that (1477, 4744) is not acceptable as genuine. More examples are provided in **Table 1**.

Although the ambiguity remains unresolved if

$$(a,b) = (uvu, xyx), \tag{5.1}$$

the probability of such occurrence is extremely minute for large p applied in public key cryptography.

6. Gaussian Generators

Definition 4: An integer G = (a, b) is a Gaussian generator modulo p if $m = p^2 - 1$ is the smallest integer for which holds that $G^m \bmod p = (1,\ 0)$.

Definition 5: An integer (a, b) is called a Gaussian generator if

$$ord\ (a,b) = p^2 - 1 \tag{6.1}$$

Proposition 6: If

$$(a,\ b)^{(p+1)/4} \bmod p = \{(\pm e,\ 0) \text{ or } (0,\ \pm f)\}, \tag{6.2}$$

then (a, b) is *not* a generator [5].

Indeed, Equation (6.2) follows from observations that

$$ord\ (a,\ b) = ord\ (e)(p+1)/4 \le (p^2 - 1)/4 \tag{6.3}$$

and that

Table 1. Resolution of ambiguity.

p	**(*a*, *b*)**	**(*A*, *B*)**	**(*B*, *A*)**
6221	(474,147)	($\underline{4}$74**4**,14$\underline{7}$**7**)	rejected
6277	(373,254)	($\underline{3}$73**3**,25$\underline{4}$**4**)	rejected
6277	(474,153)	($\underline{4}$74**4**,15$\underline{3}$**3**)	rejected
7477	(232,515)	($\underline{2}$3$\underline{2}$**2**,$\underline{5}$1$\underline{5}$**5**)	($\underline{5}$1$\underline{5}$**5**,$\underline{2}$3$\underline{2}$**2**)
9743	(545,428)	($\underline{5}$45**5**,42$\underline{8}$**8**)	rejected
9743	(727,246)	($\underline{7}$27**7**,24$\underline{6}$**6**)	rejected

$$\begin{aligned} ord\ (a,\ b) &= ord\ (0,\ \pm f)(p+1)/4 \\ &= \left[2ord\left(p - 2f^2,\ 0\right)\right](p+1)/4 \\ &\le \left(p^2 - 1\right)/2 \end{aligned} \tag{6.4}$$

Proposition 7: Let $k > 1$ be the smallest integer, for which holds

$$(a,b)^k \bmod p = (f,\ \pm f). \tag{6.5}$$

If $k < (p+1)/4$, then (a, b) is *not* a generator otherwise, if $k = (p+1)/4$ and

$$ord\ \left(p - 4f^4\right) = p - 1, \tag{6.6}$$

then (a, b) is a generator. Indeed, (6.6) follows from observation that

$$\begin{aligned} ord\ (a,\ b) &\le 4ord\left(p - 4f^4\right)(p+1)/4 \\ &= (p-1)(p+1) = p^2 - 1. \end{aligned} \tag{6.7}$$

***Remark* 5**: If $(a,b)^{(p+1)/2} \bmod p = (f,\ 0)$, then ($a$, b) is not a generator.

***Corollary* 3**: If (p + 1) / 4 is a prime,

$$(a,b)^{(p+1)/4} \bmod p = (f,\ \pm f),$$

and $ord\ \left(4f^4\right) = p - 1$, then ($a$, b) is a generator modulo prime p.

7. Conclusion

Cryptosystem based on extraction of square roots modulo composite integer n = pq of complex numbers with integer components {called Gaussian integers} is introduced and its validity is demonstrated. Security of such cryptosystem is based on complexity of root extraction if large integer factors p and q of n are not known to an intruder. The algorithm provided in this paper requires $O(\log p)$ time complexity for decryption. It is also described how to use properties of Gaussians in order to find generators that can become instrumental in the design of various cryptosystem. When this paper has been in typesetting process, Alexey Koval has suggested a constructive idea [12] that eliminates the ambiguity in case (5.1).

8. Acknowledgements

I express my appreciation to R. Rubino, M. Linderman, E. A. Verkhovsky, M. Sikorski and anonymous referees for suggestions that improved this paper.

9. References

[1] C. F. Gauss, “Disquisitiones Arithmeticae,” English Trans-

lation, Springer-Verlag, New York, 1985.

[2] A. N. El-Kassar, R. A. Haraty, Y. A. Awad and N. C. Debnath, “Modified RSA in the Domains of Gaussian Integers and Polynomials over Finite Fields,” *Proceedings of the ISCA* 18*th International Conference on Computer Applications in Industry and Engineering*, Honolulu, 9-11 November 2005, pp. 298-303.

[3] A. El-Kassar, M. Rizk, N. Mirza and Y. Awad, “ElGamal Public-Key Cryptosystem in the Domain of Gaussian Integers,” *International Journal of Applied Mathematics*, Vol. 7, No. 4, 2001, pp. 405-412.

[4] A. Karatsuba and Y. Ofman, “Multiplication of Many-Digital Numbers by Automatic Computers,” *Doklady Akademii Nauk SSSR*, Vol. 145, No. 2, 1962, pp. 293-294.

[5] B. Verkhovsky, “Enhanced Euclid Algorithm for Modular Multiplicative Inverse and Its Cryptographic Application,” *International Journal of Communications*, *Network and System Sciences*, Vol. 3, No. 12, 2010, pp. 901-906. doi:10.4236/ijcns.2010.312123

[6] L. Euler, “Vollstandige Anleitung zur Algebra,” Reclam Publishing House, Leipzig, 1941, pp. 1-527.

[7] F. Lemmermeyer, “Reciprocity Laws: From Euler to Eisenstein,” Springer-Verlag, New York, 2000.

[8] B. Verkhovsky and A. Koval, “Cryptosystem Based on Extraction of Square Roots of Complex Integers,” In: S. Latifi, Ed., *Proceedings of* 5*th International Conference on Information Technology*: *New Generations*, Las Vegas, 7-9 April 2008, pp. 1190-1191.

[9] H. Garner, “The Residue Number System,” *IRE Transactions on Electronic Computers*, Vol. EC-8, No. 2, 1959, pp. 140-147. doi:10.1109/TEC.1959.5219515

[10] M. Rabin, “Digitized Signatures and Public-Key Functions as Intractable as Factorization,” MIT/LCS Technical Report, TR-212, 1979.

[11] B. Verkhovsky, “Cubic Root Extractors of Gaussian Integers and Their Application in Fast Encryption for Time-Constrained Secure Communication,” to appear in *International Journal of Communications*, *Network and System Sciences*, Vol. 4, No. 4, 2011.

[12] A. Koval, Private Communication, 9 March 2011.