Journal of Mathematics and Statistics 4 (1): 15-20, 2008
ISSN 1549-3644
© 2008 Science Publications

Rounding Theorem the Possibility of Applying the Cryptosystems on the Decimal Numbers

Rand Alfaris, Muhamad Rezal Kamel Ariffin and Mohamed Rushdan Md Said
Institute For Mathematical Research,
University Putra Malaysia (UPM), 43400, Serdang, Selangor Darul Ehsan, Malaysia

Abstract: The possibility of using the decimal numbers in cryptography as the base of the encryption and decryption is proposed by introducing the "Rounding Theorem". Until now, cryptosystems are used the integer numbers to avoid the fractions of the decimal numbers. That is to avoid losing the original plaintext after the decryption. This theorem is proved that there is no lost for any bit of information during communications when using a cryptosystem that is based on decimal numbers. The purpose of moving from integer numbers to decimal numbers is the decimal numbers are much faster during calculations than the integer numbers. We design a new function called "The Rounding off Function", which plays the primary role in proving this theorem. The security of using the decimal numbers in cryptosystems is studied and analyzed.

Key words: Rounding off function, rounding theorem, decimal numbers

INTRODUCTION

This paper is introduced that there is no losing of the information when the cryptosystem depends on decimal numbers. Using the decimal numbers in cryptography gives the risk of losing some information of the plaintext (as an integer number) during the encryption and the decryption; as we all know that the decimal numbers are not specific numbers as the integers. There is always a fraction in decimal numbers, especially when we talk about the *recurring* type such as 0.4747474747... and the *nonrecurring* or *not exact* type such as *pi*. Despite this fact, the questions are, if the plaintext is an integer number then, is there possibility for constructing a secure cryptosystem depends on the decimal numbers and which concept can control the fraction of the decimal numbers?
Mathematically, if we have the value $43567 * \left(\frac{0.82519\cdots412}{0.82519\cdots412}\right)$, then the result would be 43567. We want to discuss the result of this value when it should be done by the computer. Therefore, if we define the result of $43567 * \left(\frac{0.82519\cdots412}{0.82519\cdots412}\right)$ as integer then the computer program would give 43567 exactly. However, if we give order to the computer program to define the same value as real number then the result would be either as $43567.123680655\cdots$ or as $43566.78659205\cdots$, that means there would be some fractions. But, in either case of the real number definition, the rounding off will handle the calculations and put it as 43567 exactly.
Our observation is, these fractions cannot change the mathematical result. The question is, why these differences in the result should stay under the controlling of the theoretical result? In our example, why the computer cannot gives the result either as $43567.8976120655\cdots$ or as $43566.10436205\cdots$ where it is obvious, for these two numbers, the result after the rounding off would be either 43568 or 43566 respectively, which they are not the same as the mathematical (theoretical) result, there would be some differences These differences lie in the interval (0,1). Here, we prove that these "differences" cannot overstep out of the controlling of the mathematical result and we prove that it will be not possible for the computer to give the last two numbers through introducing a theorem called "Rounding theorem". For this concept, we design a function called "Rounding off function" that plays the main role of proving this theorem.
Our concern relates to involve the decimal numbers in the cryptography field, because of the advantages for the security and the speed of the communications regarding this kind of numbers. For instance, when we want to calculate the value 2^{456784}, then the time needed is more than 394000 millisecond. From the other hand, and on the same processor, the time needed

Corresponding Author: Rand Alfaris, Institute for Mathematical Research, University Putra Malaysia (UPM), 43400, SSerdang, Selangor Darul Ehsan, Malaysia Tel: +60176876614

to calculate the value $2^{0.456784}$ is less than 1 millisecond. According to this fact, we can compare between using the large prime and integer numbers and using decimal numbers, even if both of the two different types of the numbers have the same key size.

The theory of the security of any cryptosystem that is based on the decimal numbers, it depends on the properties of this kind of numbers. The unknown keys are decimal numbers and that means the attacks should be through using the methods of the approximation (numerically). However, the methods of the approximation will not give the exact values, the attacker needs to the exact value of the key with the same number of digits. This strong point does not exist in other cryptosystems that depend on integer numbers because the attacker does not know really how many digits he will lose during the approximation. The methods of recovering the keys in the decimal cryptosystem should depend on numerical methods then, defiantly, the attacker will face the cumulative error. Therefore, he will not get the same decimal number. From the other hand, there are no numerical methods in applying the algorithms of this new decimal cryptosystem, so we can call it as 'one way operation', the user applies the cryptosystem without losing for any bit information, but the attacker will lose some of the information because approximation.

This paper contains three sections. In the first section, we define the terms that we use to prove the Rounding theorem[2,3,4,6]. In the second section, we construct a cryptosystem depends on the decimal numbers[1]. In the third section, we discuss the security regarding the decimal numbers[5].

THE ROUNDING OFF FUNCTION

We use the term "decimal numbers" from the viewpoint of the mathematics not from the viewpoint of the computer. So that, the decimal numbers are the numbers that belong to the uncountable interval $(0,1)$ and they have the expressions $0.j_1j_2j_3\cdots$ where $\langle j_r\rangle_{r=1}^{\infty}$ is an arbitrary sequence of integers between 0 and 9.

Definition 1: Let r be any real number, denoted by $r = J_1J_2\cdots J_{n-1}J_n.j_1j_2\cdots j_{i-1}j_i\cdots$ where both of $\langle J_i\rangle_{i=1}^{n}$ and $\langle j_k\rangle_{k=1}^{\infty}$ are arbitrary sequence of integers between 0 and 9. We call the first sequence $\langle J_i\rangle_{i=1}^{n}$ the *integer part* of the real number r, which is the part before the decimal point, while we call the second sequence $\langle j_k\rangle_{k=1}^{\infty}$ the *decimal part* of the real number r, which is the part after the decimal point.

As we said in section one, mathematically, if we have such like the value: $m\cdot\left(\frac{x}{x}\right)$, with $x \neq 0$, then the result would be m, no matter what the type of the number x is. We want to discuss the result of this value when it should be doing by the computer. The very important note here is, If we define $\frac{x}{x}$ and $m\cdot\left(\frac{x}{x}\right)$ as integers then the results would be respectively 1 and m exactly. But if we give order to the computer program to define $\frac{x}{x}$ and $m\cdot\left(\frac{x}{x}\right)$ as a real numbers then the division of $\frac{x}{x}$ would not equal to 1 exactly, and the value of $m\cdot\left(\frac{x}{x}\right)$ would not equal to m exactly. There would be some difference, this difference lie in the interval $(0,1)$, and these differences disappear when we define them as an integers. Here, in this section we prove that this "differences" cannot overstep out of the controlling of the mathematical result. What we mean by '*the controlling*' of the mathematical result is, the computer calculations cannot give the result of $m\cdot\left(\frac{x}{x}\right)$ in a way that makes the *decimal part* of the number and the *last digit* of its *integer part* change the exact value m after applying the rounding off on the number.

Definition 2: Let $R:(0,1)\to\{0,1\}$ be a function given by

$$R(0.j_1j_2\cdots j_{k-1}j_k\cdots) = \begin{cases} 0, & j_1 < 4 \\ 1, & j_1 \geq 5 \text{ or when } j_1 \geq 4 \text{ and } j_2 \geq 5 \end{cases}$$

$\forall 0.j_1j_2\cdots j_{k-1}j_k\cdots \in (0,1)$, where $j_i \in Z, i = 1,2,\cdots$ We call R the *rounding off function*.

It is easy to check that *R* is well-defined function.

Definition 3: For any real number r, we mean by Round(r) is the integer number that resulted after applying the rounding off on r.

So, if $r = J.j_1 j_2 \cdots j_{k-1} j_k \cdots$ where J is the integer part and $j_1 j_2 \cdots j_{k-1} j_k \cdots$ is the decimal part of the real number *r* respectively, then

$$\mathrm{Round}(r) = \begin{cases} J, \text{ when } R(0.j_1 j_2 \cdots j_{k-1} j_k \cdots) = 0 \\ J+1, \text{ when } R(0.j_1 j_2 \cdots j_{k-1} j_k \cdots) = 1 \end{cases},$$

where $R(.)$ is the rounding off function.

Theorem 1: (Rounding theorem)
Let m be any integer number and let *x* be any decimal number belong to the interval $(0,1)$, then the computer calculations give

$$\mathrm{Round}[m \cdot (x/x)] = m \tag{1}$$

if and only if the number $m \cdot (x/x)$ takes the form of either

i- $m + 0.j_1 j_2 \cdots j_{r-1} j_r \cdots$, where $R(0.j_1 j_2 \cdots j_{r-1} j_r \cdots) = 0$, or;
ii- $(m-1) + 0.h_1 h_2 \cdots h_r \cdots$, where $R(0.h_1 h_2 \cdots h_r \cdots) = 1$.

Proof
We have $\mathrm{Round}[m.(x/x)] = m$. We want to prove the number $m \cdot (x/x)$ gives the same theoretical result only if condition (i) or (ii) is satisfied. The proof will be by using the contradiction.

1- In condition (i) assume that $R(0.j_1 j_2 \cdots j_{r-1} j_r \cdots) = 1$. Then according to the definition (3), Eq. (1) would be $m + 1 = m$ which gives a contradiction.

and,

2- In condition (ii) assume that $R(0.h_1 h_2 \cdots h_{r-1} h_r \cdots) = 0$. Then by applying the definition (3), Eq. (1) would be $m - 1 = m$ which gives a contradiction again.

From the other side, it is obvious if $m \cdot (x/x)$ has the form either in condition (i) or (ii) then $\mathrm{Round}[m \cdot (x/x)] = m$.

Rounding theorem concludes that the decimal part that occurs by the computer calculation cannot change the theoretical results as an integer number.

For example, we choose three integer numbers: $m = 234566789876, 687574396791, 123454368653$. For any decimal number x, we give the order to the computer to calculate the value of $m \cdot (x/x)$ many times for each number, each time with different x, and we define the output two times, as integer (that means applying the rounding off) and as real number (that means the number as it is). The table below shows the results:

Table 1: computer calculations are matched with the theoretical calculations

$m \cdot \left(\frac{x}{x}\right)$, for different x	$\mathrm{Round}\left[m\left(\frac{x}{x}\right)\right]$
234566789876.0000401890808430592	234566789876
234566789876.0000120415831719936	234566789876
234566789875.9999838940855009280	234566789876
687574396791.0000701068806193152	687574396791
687574396790.9999675488383329359	687574396791
687574396791.0001826968713035776	687574396791
123454368653.0000077198064615424	123454368653
123454368653.0000077198064615424	123454368653
123454368652.9999936460576260096	123454368653

THE DECIMAL CRYPTOSYSTEM

Of course, if we want to design a cryptosystem that depends only on the decimal numbers, then we must design it such that it will not depend on group structure, finite field, discrete logarithm or prime numbers. We construct the cryptosystem by using nonlinear function; this function produces the decimal numbers under this cryptosystem. We use the notation J(.); we will call it Jay function to represent this function. All the keys in this cryptosystem are decimal numbers except the plaintext, which is an integer number.

This cryptosystem is symmetric and asymmetric, it depends on a sharing secret key between the users and each one of the users has their private key to construct his public key by using the main equation of the cryptosystem, which is:

$$Y = (J(g))^x \tag{2}$$

where, g is the secret key, x is a random decimal number as the private key and Y is the public key. x and Y are changed every block message. That mean the decimal cryptosystem is depended on one time key.

Then we construct the encryption and decryption algorithms by assisting of what we call it the encryption key: k which is also random decimal number; so the encryption algorithms are:

$$C_1 = (J(g))^k \text{ and } c_2 = Y^k * m \tag{3}$$

where, c_1 is decimal number, c_2 is a real number and m is an integer number which represent the plaintext. Then we build the decryption Algorithm as follows:

$$m = c_1^{-x} c_2 \quad (4)$$

Theorem 2: In the decimal cryptosystem, if the decryption algorithm is $m = c_1^{-x} c_2$ then, mathematically, m is the same plaintext before the encryption.

Proof: When the user does the calculation of Eq. 4, then he gets the message m because:

$$\begin{aligned} c_1^{-x} c_2 &= ((J(g))^k)^{-x} * Y^k * m \\ &= ((J(g))^x)^{-k} * Y^k * m \end{aligned} \quad (5)$$

by using Eq. 2, we get,

$$\begin{aligned} c_1^{-x} c_2 &= Y^{-k} * Y^k * m \\ &= m \end{aligned} \quad (6)$$

This proof should be enough no matter what the type of the numbers are, because the left hand side of the equation is equal to the right hand side.

However, the issue here is not that simple because the calculations have been done by the computer program, there would be fraction, so that we want to make sure that the plaintext before the encryption is the same after the decryption. This point is actually an advantage for the security of the system because the attacker will never know how many digits after the decimal point he will lose after applying any backward substitution. Below we will prove that even with using the computer program, then the decryption algorithm (4) still gives the same integer plaintext, but before we give this prove we have to say that the nature of decimal cryptosystem depends on using the decimal numbers but the calculations is not done numerically or iteratively. That means, there is no numerical method involved with this cryptosystem and each number is used only one time for each block message, so that, there is no cumulative error on the algorithms of this cryptosystem.

Proposition: In the decimal cryptosystem, if the decryption algorithm is $m = c_1^{-x} c_2$ then the computer program gives m as the same plaintext before the encryption.

Proof: Recall the decryption algorithm:

$$m = c_1^{-a} c_2 \quad (4)$$

By using theorem (2) we get:

$$c_1^{-a} c_2 = m(A^{-k} A^k)$$

The calculations of the computer give the number $c_1^{-a} c_2$ as a real number; it means have the integer part and the decimal part.

Now by applying Rounding theorem on the equation above we will get:

$$\text{Round}\left[m(A^{-k} A^k) \right] = m ,$$

if and only if

$$\begin{aligned} c_1^{-a} c_2 &= m + 0.j_1 j_2 \cdots j_{r-1} j_r \cdots \text{where} \\ R(0.j_1 j_2 &\cdots j_{r-1} j_r \cdots) = 0 \end{aligned} \quad (7)$$

or

$$\begin{aligned} c_1^{-a} c_2 &= (m-1) + 0.h_1 h_2 \cdots h_{r-1} h_r \cdots \text{where} \\ R(0.h_1 h_2 &\cdots h_{r-1} h_r \cdots) = 1 \end{aligned} \quad (8)$$

That means after the decryption we will get the same integer number that represents the plaintext m.

For example, we give three messages 32541276897621, 67119840812567 and 33681209794882 as a plaintext, we will see below two decryption texts for each plaintext because we apply Eq. 4 separately by using different encryption keys. The advantage of doing this approach is to show that the different decimal parts still give the same plaintext.

Table 2: The decryption algorithm gives the same plaintext

Decryption text	Output text
32541276897621.00260110963784	32541276897621
32541276897620.99539530885990	32541276897621
67119840812567.00772929190656	67119840812567
67119840812566.99331773334745	67119840812567
33681209794882.00868674765136	33681209794882
33681209794881.99427518909312	33681209794882

THE SECURITY

The security of this new cryptosystem depends on new theory from the viewpoint of the cryptography. As we mentioned in section one, this cryptosystem does not depend on group structure, finite field or discrete logarithmic equation and most importantly, it does not depend on the integer numbers. It depends on nonlinear equation and the concept of the decimal numbers. Practically, the methods of the attack will fail because they will face the dealing with numerical methods to recover the unknown keys because of the nonlinear equations and the decimal numbers, which is the thing that will produce the 'cumulative' error.

The size of the key: For some cryptosystems that are depending on group structure and/or finite field; there is no doubt that the plaintext before the encryption is the same plaintext after the decryption exactly. Also for these cryptosystems, the attacker knows exactly what numbers that he should trial, so that they trend to use a huge size for the keys to increase the possibilities of solving the algorithms that depend on these keys. However, these cryptosystems will be compromised once we have high-speed computers.

For the decimal cryptosystem, the attacker has the problem of what exactly the numbers that he should trial. Even if we fixed the digits after the decimal point then still the difficulty in following the numbers that picked randomly by the computer itself and also, the difficulty in following the numbers that resulted from applying the algorithms of this cryptosystem. According to this fact, this theory of the security does not really depend on the size of the key because regardless of the time needed for doing the calculations, the decimal cryptosystem depends on decimal numbers between 0 and 1 which they belong to uncountable set of numbers. Therefore, increasing the size of the key will not increase or decrease the security of the cryptosystem.

The security of algorithms: We have explained in details in section two how the decryption algorithm gives the same plaintext because it is an integer number. If the plaintext was not integer number, then definitely we cannot apply the decimal numbers in cryptography, because as we saw, if we put the plaintext in Eq. (4) as a real number then it will not give the same integer plaintext integer number, unless we apply the rounding theorem on it.

If the attacker is working with 'numerical methods field' then he can accept a 'small error'. But, he is working with the 'cryptography field' so that he cannot accept any error. This fact will protect the keys of the decimal cryptosystem.

Attack 1: Theoretically, the 'Brute-Force attack' can break the decimal cryptosystem. Practically, this attack will fail because it should deal with the imprecise decimal numbers under the "backward substitution" and this will produce "cumulative and truncation error".

Brute-force attack is typically a known-plaintext attack. Now, the attacker will begin with backward substitution by using this information together with the algorithms of the decryption and the public key, so that he will get the private and then the secret keys. Table (3) gives the results of applying this attack, by following the algorithms in section two, the attacker does not get the exact keys, a small error in the first step produce a big error in the last step. We have to mention here that we are using 9 digits after the decimal point just for the purpose of the example. Therefore, the steps of the attack will be:

Given the plaintext m: Apply this information on Eq. (4) recover the private key will produce $a+\varepsilon_1$ where a is private key and ε_1 is the first error. By using backward substitution again on the public keys algorithms to recover the secret key will produce $g+\varepsilon_2$ where g is secret key and ε_2 is the second cumulative error. This time, ε_2 is not 'small' error.

Table 3: The numerical and the exact keys under Brute-Force attack

Numerical private key	Exact private key	Numerical secret key	Exact secret key
0.400815612	0.400815615	0.528707461	0. 528276284
0.011651216	0.011651225	0.584109290	0.584332768
0.264849994	0.264857933	0.044618588	0.044648984
0.586213572	0.586213551	0.507733388	0.507734815
0.887074006	0.887074043	0.551950283	0.551851077

This attack shows us that the attacker cannot apply the rounding theorem on a and g because they are not integer numbers, so that the errors ε_1 and ε_2 will not be zero. Moreover, the relation between the error and the number of digits of the decimal number is directly proportional and finally, the trial and error method might be helpful for the attacker to adjust the keys. Therefore, we can notice that, in the decimal cryptosystem, the attacker have to apply the attack itself and then he have to begin with the adjusting.

Attack 2: Ciphertext-only attack and chosen-ciphertext attack will fail because we change the encryption key for every block message.

Attack 3: All the algorithms of the decimal cryptosystem depend on nonlinear function and the decimal numbers, so that if the attacker wants to choose any algorithm to attack separately then he should use one of the numerical methods. Any numerical method has an error percentage, and he will come back again to the same theory in the first attack. Moreover, the attacker does not have a system of nonlinear equations to recover the unknown variables in each algorithm, which is the only way for him to get the keys without errors.

There is a claim that Newton method can find the solution of the nonlinear equation of decimal cryptosystem. We test this claim and we found that there is no Newton's for nonlinear equation with two variables unless there is a system of equations and even with that, there is no prove until now that this cryptosystem can be solving by using Newton method. Moreover, Newton method is still one of the numerical methods which will also produce error. The approximation or any (fitting method) will not work because the attacker needs exactly the same numbers, not approximated numbers.

CONCLUSION

In the field numerical methods, we can accept the error that occur during applying the methods, because this concept depending on "iterative" procedures. In cryptography, we cannot accept any error. The risk was by using the decimal numbers, we proved that there is no error could be occur in applying the new decimal cryptosystem.
We achieve this proof by putting the rounding off operation as a function, so that we could calculate exactly the effect of the decimal numbers and we could control the cases that resulted by using the encryption and decryption through this new concept.

ACKNOWLEDGMENTS

We would like to acknowledge the institute for mathematical research, UPM, Malaysia for the supporting the researchers to put this research in its final form.

REFERENCES

1. ElGamal, T., 1985. A public key cryptosystem and a signature scheme based on discrete logarithms. In IEEE Trans. Inform. Theor. IT- 31 (4): 469-472.
2. Hejhal, A.D. *et al*., 1999. Emerging Applications of Number Theory, Springer, New York.
3. Hrbacek, K. and T. Jech, 1999. Introduction to Set Theory. 3rd Edn. New York, Marcel. Dikker, Inc. Ch. 4.
4. Ireland, K. and M. Rosen, 1990. A Classical Introduction to Modern Number Theory, Springer-Verlag, New York, Ch. 3 and 4.
5. Schneier, B. 1995. Applied Cryptography. 2nd Edn. John Wiley and Sons Ltd (USA), Ch. 7.
6. Schumer, P.D., 1996. Introduction to Number Theory, Boston : PWS Pub. Co.