Error Propagation Study for Gutowitz Encryption Algorithm

Daniel-Ioan Curiac, George Serban Vremescu

Department of Control Systems Engineering, "Politehnica" University of Timisoara, Romania, curiac@aut.utt.ro

Abstract: This paper presents some aspects regarding error propagation in encryption/decryption process based on H.Gutowitz algorithm,alternate first embodiement. Solutions for avoiding these problems are also presented.

Keywords: encryption/decryption algorithm, error propagation, dynamical system

1 Introduction

The encryption method described in [1] is based on the use of dynamical systems. The main feature of these systems, relevant in H. Gutowitz approach is that they may be iterated both forward and backward in time. Iterating the logistic map forward in time needs only an initial state x^0 and a value λ to be chosen. Applying the equation defining the logistic map produces the state x^1. Continuing indefinitely, this process leads to obtaining the states x^2, x^3…. Like many dynamical systems, the logistic map is irreversible, to some states corresponding more than one antecedent state. The antecedent state x^{t-1} for a state x^t of the logistic map is given by $\left(1 \pm \sqrt{1 - x^t / \lambda}\right) / 2$. To iterate the state x^t backward in time under the logistic map, one of these two states must be chosen. The Gutowitz method uses either or both of backward or forward iteration for encryption and decryption. Backward iteration of an irreversible dynamical system creates a dynamical I/O sequence for the system, which can be used for information encryption. The question is which one of the antecedent states for a state to be used. This choice can be made either arbitrarily or according to information in an input information stream. If some of the choices are arbitrary, this leads to arbitrary details in the encoding of the message. The work of a code-breaker not in the possession of the key will be very difficult. One who does possess the key, therefore knowing the dynamical systems used to encrypt, needs only to apply the known dynamical system to the cipher text, operating each dynamical system forward in time wherever the dynamical system was operated backward in time

during encryption. In this manner, all of the information inserted in the cipher text during the encryption phase is removed, thus obtaining the original plaintext.

2 Basic Methodology

The first alternate embodiment [2] uses the logistic map as the underlying dynamical system. The cryptographic key in this embodiment comprises the parameter value λ of the logistic map, and the number of times *n* the map is applied during encryption and decryption. Encryption involves only inverse iteration, and decryption involves both forward and inverse iteration of the dynamical system.

A standard form for the logistic map defines the next state of the system in terms of its previous state by x^t by $x^{t+1} = 4\lambda x^t\left(1 - x^t\right)$ (1). Here x and λ are real numbers between 0 and 1. The two possible antecedent states x^{t-1} for each state x^t are given by $x^{t-1} = \left(1 \pm \sqrt{1 - x^t / \lambda}\right) \Big/ 2$ (2).

Encrypting any information requires defining the first state of the system x^0. The value for this first state is obtained by placing the plain information on the first position after the decimal point. The next steps consist of calculating the antecedent states according to the equation (2). At each step, the choice between the two antecedent states is made arbitrary and is recorded in the dynamical I/O by placing a 1 bit for the choice of x_-^{t-1} and a 0 bit for x_+^{t-1} . The encryption phase stops at completing the entire *n* steps of the process.

The antecedent state obtained in the last step of the encryption process is submitted to the receiver. The receiver of the cipher text can decrypt it by applying the logistic map forward in time for *n* steps. The actual information sent is recovered from the first digit after the decimal point from the information obtained in the last step of the decryption phase. To rebuild the information placed in the dynamical I/O by the sender during the encryption phase, is used the inverse iteration. At each forward iteration, the state x^{t+1} is computed from the state x^t. By inverse iteration, the antecedent states of the x^{t+1} are computed and compared each with x^t. If the x_-^{t-1} antecedent state equals x^t, then the receiver knows that a 1 bit was placed in the dynamical I/O. If the x_+^{t-1} antecedent state equals x^t, than a 0 bit was placed in the dynamical I/O. By forcing certain choices for the antecedent states for some steps – known by both parties – in the encryption phase is obtained the identification information which can – for instance – give the number of the block being encrypted, to be used during checks for transmission errors.

In table 1 there is an example of the encryption/decryption process carried out for the digit 8. The first state is encoded as "0.8". The number of steps the process is performed is 8 and the second element of the key is the value of λ=1. The identification information is placed at steps 6 – 8 in the encryption phase and recovered at steps 7 – 5 in the decryption phase.

Encryption			Decryption		
Step	**Dynamical I/O**	**State**	**Step**	**Dynamical I/O**	**State**
0		0.8000000000	0		0.1166039924
1	1	0.2763932023	1	1	0.4120300054
2	0	0.9253254042	2	1	0.9690451202
3	1	0.3633667355	3	0	0.1199867010
4	0	0.8989465078	4	1	0.4223595703
5	1	0.3410554404	5	1	0.9758878547
6	1	0.0941229990	6	0	0.0941229990
7	0	0.9758878547	7	1	0.3410554404
8	1	0.4223595703	8	1	0.8989465078
9	1	0.1199867010	9	0	0.3633667355
10	0	0.9690451202	10	1	0.9253254042
11	1	0.4120300054	11	0	0.2763932023
12	1	0.1166039924	12	1	0.8000000000

Table 1. One run of the process for the alternate embodiment 1.

3 Aspects of the Error Propagation in Encryption and Decryption

The original specifications do not state whether or not the calculations imply a certain number of decimals to be used. Due to the floating point operations used during the process, when it is tried to impose a certain number of decimals to the calculation there will always be certain differences between the value that was the actual result of the operation and the value that will be used in the next step of the process. When the capacity of the programming language is used at its full potential, usually one can not go beyond the number of 15 decimals, operating with the double data type. The process with 15 decimals can not be studied nor can it not be compensated by any more operations. It can only be corrected at the end of the decryption phase. The processes that involve less than 15 decimals can be compensated by evaluating the errors involved by such a limitation.

The rounding to a number of decimal places can not be separated from the errors of rounding. These errors have a value equal to the difference between the value gained from calculations and the value passed on to the next step of the process.

This error has a known value and its sign is well determined. The question is in what way it influences the result of the next step (value and sign).

Given a formula $x = f(l_1, l_2, \ldots l_n)$ (3) where l_i are determined values with medium errors $\pm m_i$ then m_x is given by [3]:

$$m_x = \pm\sqrt{\left[\frac{\partial f}{\partial l_1} m_1\right]^2 + \left[\frac{\partial f}{\partial l_2} m_2\right]^2 + \ldots + \left[\frac{\partial f}{\partial l_n} m_n\right]^2} \quad (4)$$ where $\frac{\partial f}{\partial l_i}$ are the partial derivatives of the function f, whose numerical values are evaluated by using the average values of m_i.

This rule can be applied to the calculations implementing the first alternate embodiment, but only half of the problem is solved by this. The value of the error transmitted into the next step is known now, but its sign is yet to be discovered.

The equation used in the decryption phase is the equation (1) $x^{t+1} = 4\lambda x^t(1 - x^t)$. For easier understanding of the calculation, there must be made a replacement in the notations $x^{t+1} \Leftrightarrow y \; and \; x^t \Leftrightarrow x$, which leads to the next equation $y = 4\lambda x(1 - x)$ (5). The effect of rounding the values is expressed in the following manner: $x_c = x_f + \Delta x$ (6), where x_c is the value obtained from calculations for x, x_f is the rounded value and Δx is the difference between the two. It can be positive as well as negative.

The error involved in the evaluation of y is given by the next formula [3]: $\Delta y = 4\lambda \Delta x|1 - 2x|$ (7).

Using the equation (5), the formulas for y_c and y_f can be obtained. Δy is obtained similar to Δx, by the difference between y_c and y_f like in (8).

$$y_c = 4\lambda x_c(1 - x_c); \; y_f = 4\lambda x_f(1 - x_f); \; y_c - y_f = 4\lambda \Delta x\left|1 - (x_f + x_c)\right| \quad (8)$$

The sign of $\Delta y = y_c - y_f$ can be obtained by analyzing the terms in the equation (8):

$x_f + x_c$	$\Delta x \Leftrightarrow \Delta x^t$	$\Delta y \Leftrightarrow \Delta x^{t+1}$
≤1	≥0	≥0
	<0	<0
>1	≥0	≤0
	<0	>0

Table 2. The sign of the error for equation (1)

For the encryption process, the inverse iteration is used. The equation (2), after making the necessary simplifying notations $x^{t-1} \Leftrightarrow x \; and \; x^t \Leftrightarrow y$, can be

transcribed into $x = \left(1 \pm \sqrt{1 - \frac{y}{\lambda}}\right) \Big/ 2$ (9). The error for calculating [3] x is given by $\Delta x = \left| \Delta y \Big/ \left(4\lambda \sqrt{1 - \frac{y}{\lambda}}\right) \right|$ (10). The rounding errors are expressed by $y_c = y_f + \Delta y$ (11), with similar meanings as the equation (6) above. The difference between x_c and x_f is given by the next equation:

$$x_c - x_f = \left(1 \pm \sqrt{1 - \frac{y_c}{\lambda}}\right) \Big/ 2 - \left(1 \pm \sqrt{1 - \frac{y_f}{\lambda}}\right) \Big/ 2 = \left(\pm \sqrt{1 - \frac{y_c}{\lambda}} \mp \sqrt{1 - \frac{y_f}{\lambda}}\right) \Big/ 2 \qquad (12)$$

Two situations are evident:

I. $x_c - x_f = \left(\sqrt{1 - \frac{y_c}{\lambda}} - \sqrt{1 - \frac{y_f}{\lambda}}\right) \Big/ 2$ with the two alternatives:

a) $y_c \geq y_f \Rightarrow \frac{y_c}{\lambda} \geq \frac{y_f}{\lambda} \Rightarrow \sqrt{1 - \frac{y_c}{\lambda}} - \sqrt{1 - \frac{y_f}{\lambda}} \leq 0 \Rightarrow x_c - x_f \leq 0$

b) $y_c < y_f \Rightarrow \frac{y_c}{\lambda} < \frac{y_f}{\lambda} \Rightarrow \sqrt{1 - \frac{y_c}{\lambda}} - \sqrt{1 - \frac{y_f}{\lambda}} > 0 \Rightarrow x_c - x_f > 0$

II. $x_c - x_f = \left(-\sqrt{1 - \frac{y_c}{\lambda}} + \sqrt{1 - \frac{y_f}{\lambda}}\right) \Big/ 2$ with the two alternatives:

a) $y_c \geq y_f \Rightarrow \frac{y_c}{\lambda} \geq \frac{y_f}{\lambda} \Rightarrow -\sqrt{1 - \frac{y_c}{\lambda}} + \sqrt{1 - \frac{y_f}{\lambda}} \geq 0 \Rightarrow x_c - x_f \geq 0$

b) $y_c < y_f \Rightarrow \frac{y_c}{\lambda} < \frac{y_f}{\lambda} \Rightarrow -\sqrt{1 - \frac{y_c}{\lambda}} + \sqrt{1 - \frac{y_f}{\lambda}} < 0 \Rightarrow x_c - x_f < 0$

The sign of Δx is determined similar as for the equation (1):

x or x^{t-1}	$\Delta y \Leftrightarrow \Delta x^t$	$\Delta x \Leftrightarrow \Delta x^{t-1}$
$x = \left(1 + \sqrt{1 - \frac{y}{\lambda}}\right) \Big/ 2$ *or* $x^{t-1} = \left(1 + \sqrt{1 - \frac{x^t}{\lambda}}\right) \Big/ 2$	≥0	≤0
	<0	>0
$x = \left(1 - \sqrt{1 - \frac{y}{\lambda}}\right) \Big/ 2$ *or* $x^{t-1} = \left(1 - \sqrt{1 - \frac{x^t}{\lambda}}\right) \Big/ 2$	≥0	≥0
	<0	<0

Table 3. The sign of the error for the equation (2).

3 Methods of Avoiding Error Propagation

Though applying the method described above resolves some of the problems caused by rounding to a certain number of decimal places, it is not perfect. The calculation is still affected by the rounding errors due to the limitation of the programming language used to implement the method (usually one can use a maximum number of 15 decimal places when using the double data type). Therefore, it is necessary to apply a certain correction for an useful result. Further more, even when there is no limit regarding the number of decimal places, the correction must take place at the end of the decryption process, due to the same reason mentioned above.

The following table presents the values of these corrections and the possible parameters given to the system in order to provide a secure encryption of the information. The method described by Gutowitz states that the plain text is recovered from the first position after the decimal point. This is equivalent to getting the integer portion of the number obtained by multiplying by 10 the result of the decryption process.

No. of Steps	Correction for standard process	Correction for the rounding process	Number of decimals for rounding
1	3.99680288865056E-15	2.99299474093573E-12	12
2	1.69864122767649E-14	2.36530239838828E-11	12
3	1.74027459109993E-13	1.71925973457832E-10	12
4	1.02101660459653E-12	1.04501296505077E-09	12
5	7.50070838773098E-12	8.36303704065956E-09	12
6	5.67530467066035E-11	6.94935329836888E-08	12
7	4.69110611467372E-10	6.10516498023017E-07	12
8	4.03960298545059E-09	4.26269194503393E-06	12
9	2.8577477051428E-08	2.48605161110027E-05	12
10	2.49921816020127E-07	1.88613482842701E-04	12

Table 4. The values of the corrections

Each value of the corrections presented for the rounding process works only for a number of decimals of 12 and above. If the process is implemented in Visual Basic, this means that the domain for the number of decimals this process works is between 12 and 14 decimals. The case with 15 decimals will not be considered because 15 is the maximum number of decimals possible for the operations in Visual Basic using the double data type and rounding to 15 decimals is identical to using the standard process.

A more specific values for the corrections can be determined for the processes classified by the number of steps they consist of. In every category of processes, according to the number of decimals used, there can be defined a value for the

correction. As seen in table 5, this leads to a larger domain of number of decimals for which the process works. The difference mentioned in the table is the result of a substract operation between the encoded form of the number encrypted and the result of the decryption process. The value for the correction is the value of the maximum difference for the number of steps and decimals used in the process. The value must be chosen so that, when applied, it gives a good result even for the cases when the difference is the minimum difference. This means that, for the process in 3 steps and 3 decimals, the correction can not be used, but is can be used for a process in 3 steps and 4 decimals.

	Encryption in 3 steps				**Encryption in 9 steps**			
Decimals	**Total**	**Succeed**	**Minimum difference**	**Maximum difference**	**Total**	**Succeed**	**Minimum difference**	**Maximum difference**
1	800	46	-45.3811	0.9	51200	2414	-6.8E+161	0.9
2	800	162	-1.4752	0.9	51200	1406	-5.45E+73	0.9
3	800	414	-0.16493	0.2	51200	1692	-3.69E+27	0.9
4	800	414	1.22E-05	0.01611	51200	3550	-4.48E+13	0.9
5	800	440	3.58E-07	0.00167	51200	7218	-7.27E+5	0.9
6	800	426	4.60E-08	1.46E-04	51200	13236	-97.2337	0.9
7	800	454	7.26E-09	1.78E-05	51200	20780	-2.49351	0.9
8	800	432	3.72E-10	1.64E-06	51200	27172	-0.340681	0.3
9	800	424	6.58E-11	1.51E-07	51200	27662	1.09E-10	0.03513
10	800	396	8.67E-12	1.81E-08	51200	27556	7.14E-11	3.18E-03
11	800	436	1.13E-13	1.68E-09	51200	27706	3.84E-12	2.24E-04
12	800	434	3.41E-14	1.72E-10	51200	27961	2.40E-14	2.49E-05
13	800	437	2.00E-15	1.13E-11	51200	27570	4.00E-15	2.49E-06
14	800	425	9.99E-16	1.71E-12	51200	27516	9.99E-16	2.97E-07

15	800	440	4.02E-16	1.77E-13	5120 0	27809	9.99E-16	2.94E-08

Table 5. The maximum difference for processes of 3 and 9 steps

The 3 steps process has a correction of 1.51E-7 for 9 decimals and the 9 steps process has a correction of 0.03513 for the same number of decimals. This means that, as the number of steps increases, it is more difficult to correct the process for lower number of decimals, so, it is best to chose a higher number of decimals when running a process with higher number of steps. Beside these, the process does not work in all the situations when λ equals to 1 and the number encrypted is 0. These situations lead to overflow errors which cancel the execution of the encryption/ decryption process process.

Determining the values for the corrections can be very difficult. The values presented above were determined by running the system for every possible combination of the dynamical I/O.

A way of avoiding the evaluation of the corrections is to perform a process of rounding to 0 decimal places on the number obtained after multiplying the result of the decryption phase by 10. The corrections are no longer needed when the plain text is recovered by a round operation. This improves greatly the efficiency of the process, avoiding the entire process of evaluating the corrections needed for the encryption/ decryption process. For the standard process – that does not involve any limitation of the number of decimals used – the only errors that appear are in some of the situations described above, when the number encrypted is 0 and λ equals 1.

The process that forces the calculation to use rounding to a number of decimal places is significantly improved compared to the previous method of obtaining the plain text.

When the process consists of one step only and one decimal place, the calculations performed are generally wrong. All the other cases, implying a greater number of decimals, work. Along with the increase in the steps of the process, the lower limit of the number decimals at which the limitation is done without causing failure to the process increases.

A drawback of this method is the dramatic increase of the maximum difference between the value of first state of the encryption phase and the value of last one of the decryption phase – which, at east in theory, should be equal. Its behavior can be observed from the tables below.

Decimals	**All**	**Succeeded**	**Failed**	**Maximum difference**	**Overflow**
1	200	54	146	0.3	0
2	200	200	0	Not available	0
3	200	200	0	Not available	0
---	---	---	---	---	---

15	200	200	0	Not available	0

Table 6. Values of the maximum difference for the process of 1step

Decimals	All	Succeeded	Failed	Maximum difference	Overflow
1	25600	1230	24370	0.9	1892
2	25600	686	24914	0.9	764
3	25600	1170	24430	0.9	374
4	25600	3462	22138	0.9	270
5	25600	7874	17726	0.9	254
6	25600	14576	11024	0.9	254
7	25600	22566	3034	0.5	254
8	25600	25346	254	Not available	254
---	---	---	---	---	---
15	25600	25346	254	Not available	254

Table 7. Values of the maximum difference for the process of 8 steps

It grows from 0.3 in the case of the process in 1 step and calculations with only one decimal to 0.9 for the process in 8 steps and number of decimals between 1 and 6. This means that, with the growth of the number of steps, it is not suitable to our purpose to obtain the plain text by a rounding process when a specific number of decimals is forced upon the process. This method can be useful only when the calculations are free from such a requirement. Using the first method – that does not imply the use of rounding in obtaining the original information – then the domain of the number of decimals is determined by the corrections chosen.

The conclusion that can be drawn from these is that the solution for this problem is a mixed method. When it is needed to perform the process with a number of decimals, the plaintext will be obtained from the integer portion of the product between the result of the decryption phase and 10, which implies obtaining the suitable corrections, and, when such a need is not desired, the plaintext is the integer portion of the rounded to 0 decimals product mentioned above.

4 The Encryption/Decryption Improved Methodology

The cryptographic key of the modified method includes the parameter λ, the number of steps of the process and the values of the corrections that might be applied after the decryption phase. Both parties of the encrypted communication share this information. They also have to agree upon the number of decimals used during the encryption. Each step of the encryption phase consists of several operations and is applied upon the number gained from the previous step, except the first one, who is applied directly upon the plaintext, after it has been encoded

as necessary (the number encrypted is placed in the first position after the decimal point, the integer portion is equal to 0 and all other decimals are 0 as well).

First, the number from the previous step is rounded, if needed, to the specified number of decimals. The error caused by this rounding is evaluated, then it is evaluated the error of the calculation, according to the error propagation rule, described in the equation (7). The number to be sent to the next step is then obtained, as it is presented in the equation (2), and modified accordingly, with the error evaluated previously. After all the steps of the encryption phase are passed through, a last rounding is performed, if it was agreed upon between the two parties to perform the calculations to a specific number of decimals, then the information is passed to the receiver. The receiver performs the decryption process, which is the inverse of the process of the encryption phase.

Every step in the decryption phase performs at the beginning a rounding operation upon the number received from the previous step, respectively the encryption phase, for the first step. The next operation is to evaluate the rounding error that occurred, useful in obtaining the overall error of the operations in the step, as presented in the equation (10). The number that will be supplied to the next step (or the result of the decryption phase) is obtained according to the equation (2) and updated with the overall error. When the number of steps has been reached, the process stops, the last number evaluated being the result of the decryption process. The plain text will be recovered from this result according to the way all the calculations were performed.

If rounding was used, then the correction has to be added to the result, for a correct answer, but the result of the decryption process is no longer rounded. The original information is then recovered from the first decimal position after the decimal point. If there were no such limitation, as it regards the number of decimals used, no correction is applied and the plain text is obtained the same as the previous way.

Conclusions

The solutions presented in this paper improve the algorithm proposed by H. Gutowitz (alternate first embodiement), solving the problems created by the limitations of the computers in the case of a software implementation, leading towards an encryption method with obvious practical use. Implementing this method with specialized hardware equipment would greatly improve the performance of this method, as well as making it more resistant to code-breakers and tampers.

References

[1] Gutowitz, H.A. – Method and Apparatus for the Encryption, Decryption and Authentication of Messages Using Dynamical Systems, U.S. Patent 5,365,589 Issued Nov. 15, 1994.

[2] Gutowitz, H.A. – Cryptography with Dynamical Systems, In: Cellular Automata and Cooperative Phenomena, Kluwer Academic Press, 1993.

[3] *** - Formular Matematic şi Tehnic pentru elevi, studenți și tehnicieni, pages 815 – 818, Editura Tehnică, 1950.

[4] Abramowitz, M. and Stegun, I. A. (Eds.). Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables, 9th printing. New York: Dover, p. 14, 1972.