One-way Functions from Chebyshev Polynomials

Kai-Yuen Cheong [*]

May 8, 2012

Abstract

In the past twenty years, the study of the conjunction of chaos and cryptography has attracted much interest but also met with many problems. Today the security of chaos-based encryptions is usually not considered comparable to those based on number theoretic functions. In this paper, instead of making an encryption system, we focus on the more fundamental notion of one-way function, which is a well-defined function that is easy to evaluate but hard to invert. We see that it is more natural to compare chaotic systems with one-way functions, and such a study could possibly give new insights for chaos-based cryptosystems. We propose a function based on Chebyshev polynomials, and we argue it is likely a one-way function.

1 Introduction

1.1 Chaos and Cryptography

The study of cryptography using nonlinear dynamics and chaotic systems began more than twenty years ago. Generally, there is a hope that the long-term unpredictability of chaotic systems can be used as a source of computational complexity in certain encryption systems. Most chaotic systems have very simple descriptions (using small amount of memory in computational terms), but their behavior cannot be predicted. This makes them promising candidates for cryptographic applications. But since the unpredictability only comes in the long run, it is not easy to be used effectively. This is inevitable since chaotic systems are usually continuous, unlike the discrete, number theoretic functions normally found in modern public-key cryptosystems.

At some point, chaos cryptography has gathered much interest in the mainstream cryptography community, but also demonstrated its difficulties. In 1991, in a major conference in cryptography, a chaotic cryptosystem has been proposed [9] and then (partially) broken [4] in very short time within the conference. Ten years after that, the interest of chaos-based cryptography has faded, and

[*]School of Information Science, Japan Advanced Institute of Science and Technology, 1-1 Asahidai, Nomi, Ishikawa, Japan. Email: kaiyuen@jaist.ac.jp

is mostly limited to the nonlinear systems community only. During this period there are also some reviews of the problems and properties of chaos-based cryptography like [11] and [7], attempting to sort out new strategies. Since then, opinions are divided while research continues. In some cases, for example, when the plaintext is itself is a continuous signal, or when a trade-off between security and system speed is required, chaos-based encryptions are still believed to be possibly advantageous.

1.2 Chaos and Chebyshev Polynomials

The fact that Chebyshev polynomials is a source of chaotic dynamics has been known for a long time. They also have an interesting commutative property. Based on such properties, a public-key encryption scheme based on Chebyshev polynomials was proposed in [13]. But the first system working on the real number domain is found to be flawed later by [3] and [5]. A symmetric-key scheme using a polynomial with similar properties [2] is also known to be insecure [1]. In [12], the Chebyshev polynomials are used in the discrete sense, working on a finite field. It is so far considered secure, but with some issues [15, 6].

In this paper, instead of a cryptosystem, we try to create a one-way function, which is a more fundamental and natural concept. In our case, the one-way property relies on the unpredictable dynamics generated by iterations of Chebyshev polynomials. Bearing similarities with [13], we try to avoid the known weakness of [13] in our proposal.

1.3 Notion of One-way Functions

One-way function is a very fundamental concept in cryptography [8]. In a one-way function, it is easy to compute the function itself, but it is hard to reverse the process to find any pre-image of a randomly given image.

The definition of one-way function is based on computational complexity theory. The existence of one-way function requires $P \neq NP$, but $P \neq NP$ does not directly imply the existence of one-way functions [17]. Since whether $P = NP$ is still a long standing problem in complexity theory, there are no proven one-way functions. Currently we can only have candidates of functions believed to be one-way. Such candidates are abundant, and they are used in cryptography practically. The most obvious examples include discrete exponentiation (as opposed to discrete logarithm), integer multiplication (as opposed to factorization).

1.4 Applications of One-way Functions

One-way functions are important to cryptography in both theory and practice. In the theoretic side, the existence of one-way function is a necessary condition for the existence of most cryptographic primitives [10]. In the practical side, it is also a sufficient condition for many common cryptographic primitives and functions. The latter fact is shown by explicit constructions of the cryptographic functions from one-way functions. Examples include symmetric-key encryptions, commitments schemes and pseudorandom generators.

When a cryptographic primitive is created based on one-way functions, an important separation of system function and security can be achieved. Due to the simple definition of a one-way function, it is often easier to verify its one-way property than to try to show the security of the whole cryptographic primitive constructed, which may have many components. Also, in case a one-way function is shown to be flawed, the cryptographic primitives using it can immediately switch to another candidate of one-way function, possibly with a slight modification of settings, and remain secure based on the new one-way function.

2 Preliminaries

2.1 One-way Functions

The definition of one-way function is straightforward [8]. For a well-defined function $f(x)$, it is a one-way function if:

- There is a polynomial time, deterministic algorithm to evaluate $f(x)$ correctly. This algorithm is known explicitly.
- For any probabilistic polynomial time algorithm A trying to invert the function $f(x)$, we have $Pr[f(A(f(x))) = f(x)] < 1/p(m)$ for every positive polynomial p, where m is the length of x as a binary string, and x is taken uniformly random.

That is, given an image $f(x)$, it is hard to find any pre-image y such that $f(y) = f(x)$. It is not necessary to find the original x. Also note that the definition of one-way function does not say anything about the difficulty to obtain partial information of the pre-image. In other words, an attacker of the one-way function is given an image z, and the attack is considered successful if and only if he finds y such that $z = f(y)$. The attack is not a success even if a few bits of y are recovered.

2.2 Chebyshev Polynomials

There are actually two kinds of Chebyshev polynomials. We only consider the more commonly known first kind. The definition of Chebyshev polynomials $T_k(x)$ is:

$$\begin{aligned} T_0(x) &= 1 \\ T_1(x) &= x \\ T_k(x) &= 2xT_{k-1}(x) - T_{k-2}(x). \end{aligned} \tag{1}$$

The first few of them that demonstrate chaotic properties are:

$$T_2(x) = 2x^2 - 1$$

$$\begin{aligned} T_3(x) &= 4x^3 - 3x \\ T_4(x) &= 8x^4 - 8x^2 + 1. \end{aligned} \tag{2}$$

In this paper, we basically only use $T_2(x)$, in order to make use of the chaotic properties with minimal computations. Chaotic properties appear only for $x \in (-1, 1)$. So we will assume that x is in this range.

The Chebyshev polynomials have many interesting properties. The following two properties would be of special interest. First, the polynomial can be expressed using cosines.

$$T_k(x) = \cos(k \cos^{-1} x). \tag{3}$$

Next, the compositions of Chebyshev polynomials are also Chebyshev polynomials. In particular,

$$T_r(T_s(x)) = T_s(T_r(x)) = T_{sr}(x). \tag{4}$$

This is known as the commutative property of Chebyshev polynomials.

3 Our Construction

As a function, the Chebyshev polynomial $T_k(x)$ is not hard to invert when k is relatively small. It is just solving an equation with a polynomial of degree k in real number domain. To make k large, we can make use of the commutative property of Chebyshev polynomials, that $T_s(T_r(x)) = T_{sr}(x)$. That is, if we apply $T_2(x) = 2x^2 - 1$ on an input repeatedly for n times, the result is $T_{2^n}(x)$. Also, if $-1 < x < 1$, it will be true that $-1 < T_k(x) < 1$ for any k. Therefore the numbers involved in every step will not go out of range. This is the core idea of our one-way function. Note that the function $T_2(x)$ is homomorphic to the logistic map $L(x) = 4x(1 - x)$, which is more popular in chaos generation [16]. An orbit of logictic map (generated by applying the map repeatedly to an initial value) can be mapped to an orbit of $T_2(x)$ by the simple function $1 - 2x$. That means the dynamics of $L(x)$ and $T_2(x)$ are the same.

3.1 Domain Issue

The discussion above suggests that we can evaluate $T_{2^n}(x)$ in just n steps of evaluating $T_2(x)$. But one problem to use it as a one-way function is that it is easy to invert the algorithm step by step. Each step involves solving x with a known $T_2(x)$. This will always give two roots $\{x', -x'\}$. The adversary can choose one randomly in order to proceed, and will finally get a candidate of the original input. Even though it is not likely the real original input, the function $T_{2^n}(x)$ has been inverted anyway, and the one-way function is broken according to the definition.

To make $T_{2^n}(x)$ hard to invert against this method, we can limit the domain for the original input x. For example, let the actual input i be a binary string (treated as a number) of fixed length m_1, we add a random key ρ of binary length m_2. Using a fixed point notation of numbers, the concatenation of i and ρ is used as the binary digits below the fixed point, with ρ being less significant than i in it. In other words, we treat i and ρ as positive integers in binary form, and define $x = i2^{-m_1} + \rho 2^{-m_1-m_2}$. The constructed number x is used as the input to the Chebyshev polynomial. It is of $m = m_1 + m_2$ bit precision. In this case, if the adversary tries to obtain just any pre-image x' for an image of the function, it is unlikely that it will match the given ρ. The set of parameters (ρ, n, m_1, m_2) can be published to specify a particular one-way function using $T_2(x)$ with this method.

3.2 Algorithm and Precision

In order to make sure that the function can be evaluated efficiently, we must limit the precision as it is a function of real numbers. The exact algorithm can be as follows. The input to the first step has m digits below the fixed point. Evaluating $2x^2 - 1$ exactly requires no more than double the space, which is $2m$ bits. Of course, we need to modify it, or the number of digits will be doubled every round. We do not need to store the sign of the number, except for the final round. This is because the sign of x is irrelevant in the next step, which only needs to know x^2. So, after expressing the number in $2m$ bits, we simply truncate the result back to m bits for the next round. This is the common approach to simulate chaotic functions with digital computers.

Repeat the procedure above for n times, we produce a number with m bits. Finally this m bit result is truncated to m_1 bits, such that the final output has the same length as the original input. We call the final new one-way function defined this way $G_n(i)$. Due to the truncation error, the result will not be the same as $T_{2^n}(x)$. It can be viewed as a noisy version of a chaotic dynamic system based on Chebyshev polynomials. Due to sensitivity to initial conditions in chaotic systems, the noisy and noiseless versions behave differently. This difference is an advantage, and is critical to security. Since $T_k(x) = \cos(k \cos^{-1} x)$, the attacker may try to recover x from $T_k(x)$ using this expression. Same as the known weakness of [13], this attack is actually possible and practical. But due to the truncations in the algorithm of our one-way function, $G_n(i)$ and $T_{2^n}(x)$ are different. In fact, any attack directly using shortcuts to invert Chebyshev polynomials will not succeed in attacking this one-way function.

For a more precise description of the one-way function, we first define the following algorithm $g(x)$:

1. Input number x. It is required that $-1 < x < 1$.

2. Ignore the sign of x and treat it as a binary number with m bits precision. That is, we compute
$$X = \lfloor 2^m |x| \rfloor / 2^m \tag{5}$$
where $\lfloor x \rfloor$ means the largest integer smaller than x for a real number x.

3. Calculate $y = 2X^2 - 1$ accurately, which can be represented in $2m$ binary digits.

4. Output y.

Next, the definition of algorithm $G_n(i)$ is as follows. It is assumed that there is a system parameter ρ of m_2 bits.

1. Input i. It is required that it is a binary string of m_1 bits. Then treat i and system parameter ρ as binary integers.

2. Calculate $x = i2^{-m_1} + \rho 2^{-m_1-m_2}$.

3. Using x as the first input, iterate $g(x)$ for n times. Set the last output to y.

4. Truncate y such that only m_1 bits remain. Output the result.

In the algorithm of $G_n(i)$, $g(x)$ is repeated n times and the final result is truncated back to m_1 bits. The last truncation step further ensures that the information given to the adversary is limited. This is reasonable because the input and output now have the same binary length. $G_n(i)$ is our final one-way function.

The evaluation of $G_n(i)$ is rather efficient as it takes n steps and each step is mainly squaring a number of m bits. The memory usage is also extremely efficient, as we only need enough space for the computation of the current stage from the previous stage. Of course, a space for step counting up to n is also required.

4 Security Issues

First, it is reasonable to set $m_1 = m_2$, due to the unavoidable brute-force attacks. If the attacker, given an image z, just randomly chooses some input i and evaluate $G_n(i)$, his success rate for $z = G_n(i)$ is in order of 2^{-m_1}. If the attacker try to invert $G_n(i)$ up to getting a candidate of x as defined in the beginning steps of $G_n(i)$, the chance that x is consistent with ρ is in order of 2^{-m_2}.

Also, we require $n \geq m$. This is necessary if we want every bit of $G_n(i)$ to be sensitive to every bit of $x = i2^{-m_1} + \rho 2^{-m_1-m_2}$. The Lyapunov exponent of $T_2(x)$ is $\ln 2$, meaning that on average a change in the least significant bit of x takes m iterations of $T_2(x)$ to affect the most significant bit of the result, where m is the number of bits in x. For a lower n, a small change in x may not produce enough change in the output. In this case, some kinds of attacks would be possible, because similar inputs would be mapped to similar outputs.

Another possible problem is concerned with the size of the actual range of our function. $T_2(x)$ is a two-to-one mapping in the real number domain. From all the possible inputs going through the iterations of $g(x)$, the number of possible outputs may decrease every time $g(x)$ is applied, because different inputs could be mapped to the same output. If the final range of the function is too small, it will not be a good one-way function. We argue that this question can be answered together with the next problem below.

With any fixed i, it is clear that $G_n(i)$ is eventually periodic in n. If we write the series $g(x), g^2(x)...$, where $g^j(x)$ is the iteration of $g(x)$ for j times, there will be some smallest k and $k' > k$ such that $g^k(x) = g^{k'}(x)$. Obviously it also means $g^{k+j}(x) = g^{k'+j}(x)$ for every j. The value of $k' - k$ is called the period of the orbit, and the series $g(x), g^2(x)...g^{k-1}(x)$ is called the transient orbit.

This is a common problem for digitized chaotic systems [14], which is inherently different from their analog counterparts. If the period is too small, the one-way function may be vulnerable. A study in [18] shows that in general the average period length is proportional to the square root of the domain size of the function. Intuitively, this observation makes sense due to the birthday paradox. If $g(x)$ is behaving like a random function, then the series $g(x), g^2(x)...$ are random numbers in the domain. A periodic orbit is found when $g(g^{k-1}(x)) = g(g^{k'-1}(x))$ but $g^{k-1}(x) \neq g^{k'-1}(x)$. This is a collision of $g(x)$ and on average it would happen when the sample size is near the square root of domain size. A more detailed study of this issue is shown in [16] for the logistic map, which automatically applies to $T_2(x)$ as well.

It means the period length increases exponentially with the precision used. Thus it is not a problem in our one-way function, as our n is much smaller than the average period. It also implies that the size of the range of our function is at least in order of the square root of the domain size. Our domain size is 2^m, and so the range size is at least in order of $2^{\frac{m}{2}}$, which is acceptable.

4.1 Numerical Illustrations

Just for example let $n = 60$. Set $m_1 = m_2 = 30$. Let the user's input be $i = 322122547$, which is smaller than 2^{30}. Let $\rho = 332859965$. Setting $x = (i2^{30} + \rho)/2^{60}$, we have

$$\begin{aligned} x &= 345876451500165693/2^{60} \\ G_n(i) &= -956954862/2^{30}. \end{aligned} \tag{6}$$

For robustness against attacks, we can first see the sensitivity of $G_n(i)$ to x. Set $\rho = 332859966$:

$$\begin{aligned} x' &= 345876451500165694/2^{60} \\ G_n(i) &= -480296176/2^{30}. \end{aligned} \tag{7}$$

Next we try to compare $T_{2^n}(x)$ and $G_n(i)$ to show that solving $T_{2^n}(x)$ is not effective to attack the one-way function. The accurate evaluation of $T_{2^n}(x)$ can be done by the cosine based form of Chebyshev polynomials, up to any desired precision.

$$\begin{aligned} x &= 345876451500165693/2^{60} \\ T_{2^n}(x) &\approx -0.7712 \\ G_n(i) &= -956954862/2^{30} \approx -0.8912. \end{aligned} \tag{8}$$

4.2 Open Challenge

Here we post a challenge. If $n = m = 128$ and $m_1 = m_2 = 64$ with $\rho = 0$, can anyone solve i in:

$$G_n(i) = 12395610133708948990/2^{64} \tag{9}$$

with any efficient algorithm that runs in significantly fewer than 2^{64} steps?

5 Conclusion

Taking $m = n$ and $m_1 = m_2$, the final family of our one-way functions using $T_2(x)$ is indexed by n and ρ only. One announces the choice of n and ρ to specify a one-way function. Due to sensitivity to initial condition of a chaotic system based on the Chebyshev polynomials, a slightly different ρ will give a very different one-way function. The security parameter is simply n and thus the security is scalable. Currently there seems to be no known method that could invert this function, and therefore we argue that it is a candidate for one-way functions.

References

[1] G. Alvarez, F Montoya, M. Romera, and G. Pastor: Cryptanalysis of an ergodic chaotic cipher, *Physics Letters A*, no.311, pp.172–179, 2003.

[2] M.S. Baptista: Cryptography with chaos, *Physics Letters A*, no.240, pp.50–54, 1998.

[3] P. Bergamo, P. D'Arco, A. De Santis, and L. Kocarev: Security of public key cryptosystems based on Chebyshev polynomials, *IEEE Transactions on Circuits and Systems I*, vol.52, no.7, pp.1382–1393, Jul 2005.

[4] E. Biham: Cryptanalysis of the chaotic-map cryptosystem suggested at EUROCRYPT'91, In *Advances in Cryptology — EUROCRYPT 1991*, LNCS 547, pp.532–534, 1991.

[5] K.Y. Cheong and T. Koshiba: More on security of public-key cryptosystems based on Chebyshev polynomials, *IEEE Transactions on Circuits and Systems II*, vol.54, no.9, pp.755–799, Sept 2009.

[6] F. Chen, X. Liao, T. Xiang, and H. Zheng: Security analysis of the public key algorithm based on Chebyshev polynomials over the integer ring Z_N, *Information Science*, vol.181, no.22, pp.5110–5118, 2011.

[7] F. Dachselt and W. Schwarz: Chaos and cryptography, *IEEE Transactions on Circuits and Systems I*, vol.48, no.12, pp.1498–1509, 2001.

[8] O. Goldreich: *Foundations of cryptography, volume I*, Cambridge University Press, 2001.

[9] T. Habutsu, Y. Nishio, I. Sasase, and S. Mori: A secret key cryptosystem by interating a chaotic map, In *Advances in Cryptology — EUROCRYPT 1991*, LNCS 547, pp.127–140, 1991.

[10] R. Impagliazzo and M. Luby: One-way functions are essential for complexity based cryptography, In *Proc. 30th IEEE Symposium on Foundations of Computer Science*, pp.230–235, 1989.

[11] L. Kocarev: Chaos-based cryptography: a brief overview, *IEEE Circuits and System Magazine*, vol.1, no.3, pp.6–21, 2001.

[12] L. Kocarev, J. Makraduli, and P. Amato: Public-key encryption based on Chebyshev polynomials, *Circuits, Systems and Signal Processing*, vol.24, no.5, pp.497–517, 2005.

[13] L. Kocarev and Z. Tasev: Public-key encryption based on Chebyshev maps, In *Proc. IEEE Symposium on Circuits and Systems (ISCAS'03)*, vol.3, pp.28–31, 2003.

[14] S. Li, G. Chen, and X. Mou: On the dynamical degradation of digital piecewise linear chaotic maps, *International Journal of Bifurcation and Chaos*, vol.15, no.10, pp.3119–3151, 2005.

[15] X. Liao, F. Chen, and K.W. Wong: On the security of public-key algorithms based on Chebyshev polynomials over the finite field Z_N, *IEEE Transactions on Computer*, vol.59, no.10, pp.1392–1401, Oct 2010.

[16] T. Miyazaki, S. Araki, Y. Nogami, and S. Uehara: Rounding logistic maps over integers and the properties of the generated sequences, *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol.E94-A, no.9, pp.1817–1825, 2011.

[17] C.H. Papadimitriou: *Computational complexity*, Addison-Wesley, 1994.

[18] D.D. Wheeler and R.A.J. Matthews: Supercomputer investigations of a chaotic encryption algorithm, *Cryptologia*, vol.15, no.2, pp.140–152, 1991.