Design Decisions and Security Considerations for Image Encryption via Chaotic Maps

Michael C. Hughes
F. W. Olin College of Engineering
(Dated: March 15, 2010)

Chaotic maps are a promising new direction for image encryption compared to conventional cryptosystems. Advantages include highly sensitive keys (e.g. initial conditions), greater run-time efficiency, and tractability within embedded imaging devices such as satellites. This investigation describes the design and justification of a simple chaotic cryptosystem based upon a revised logistic map. Two primary design objectives, which we call diffusion and confusion, are articulated. A shuffle-and-mask algorithm based on the revised logistic map is then presented which fulfills diffusion goals. Later we present a plaintext attack that breaks the presented scheme and highlights how confusion design weaknesses can be exploited.

Catalyzed by dramatic increases in both the availability of image capture devices and the bandwidth of information transfer, digital photos and video find utility in a growing number of applications. Many of these applications require confidentiality of image content, whether for medical privacy, discreet surveillance, or other uses, and so research in image encryption remains a widely-pursued and well-funded research field.

The increasing number of embedded cameras within remote sensing devices (e.g. satellites and UAVs) require special research attention due to limited computational resources. Many in-practice cryptographic schemes that are widely used in e-commerce have significant shortcomings when applied the image processing domain. RSA, DES, and other systems that secure online banking rely on desktop computing power to perform complex mathematical operations such as modular exponentiation, primality testing, and so on. Furthermore, even when processing power is not a concern these schemes offer poor run-time performance on the order of several seconds per image, which precludes real-time transfer of high-frame rate video. Additionally, conventional algorithms often cannot adequately disperse the highly correlated information found in images to maintain adequate confidentiality [1].

These disadvantages of conventional cryptography have inspired a recent wave of research applying *chaotic maps* to cryptographic applications. A chaotic mapping is a parameterized system of nonlinear equations that iteratively forces system state through a pseudo-random trajectory across bounded state space in a way that is highly sensitive to initial conditions. Many researchers have observed that this behavior is highly analogous to conventional symmetric key cryptographic schemes. That is, if two communicating parties agree on a specific map function as well as a secret set of parameters and initial conditions, the sender can use the function as well as secret parameters (keys) to produce an arbitrarily-long source of pseudo-random numbers with which to scramble a secret message. Later, the receiver can consume this scrambled message and, after generating the same exact source of pseudo-randomness, invert the scrambling operation to recover the secret message.

Chaotic encryption systems have many advantages over conventional schemes. First, although non-linear the typical mathematical operations required have considerable performance advantages compared to the algorithmic complexities and often specialized hardware required for AES and other desktop standards, making chaotic encryption favorable for cost-efficient embedded devices. Second, the sensitivity of chaotic systems to parameter values and initial conditions allow confidence in using these values as secret keys, as near-miss values will yield dramatically different pseudo-random trajectories. Finally, mathematical tools for analyzing chaotic behavior (e.g. Lyaponov numbers) are well-studied and helpful in designing an effective protocol.

Many different proposals for chaos-based encryption have been made (e.g [1], [3], [4]), and each one differs in the type, number, and dimensionality of the map used. This paper provides a modest tour of the design considerations, algorithmic execution, and security analysis of a simple one-dimensional cryptosystem based on the logistic map. Although advanced multi-dimensional systems have been proposed and analyzed (e.g. [1], [3]), high-dimensionality is not necessary and may place additional burdens on performance. We thus restrict ourselves for the sake of clarity to one-dimension, so that we may provide detailed justification of design choices for the map and parameter space while avoiding unnecessarily sophisticated mathematics.

This paper is organized into three sections. First, we describe the desirable mathematical properties of encryption systems: diffusion and confusion. Second, we examine the design decisions involved in choosing a particular map and key space for implementation so that it achieves desired behavior. Next, we outline the algorithm for a cryptographic system based upon a modified logistic map that follows from the design process previously described. Finally, we analyze the security vulnerabilities of the example scheme, showing that it upholds diffusion goals but can be completely broken due to confusion weaknesses. We close with recommendations for further improvement.

Mathematical Theory: Chaos meets cryptography — Before launching into the details of a chaos-based encryption algorithm, we should first step back and ask, "What properties make a chaotic map suitable for cryptography?"

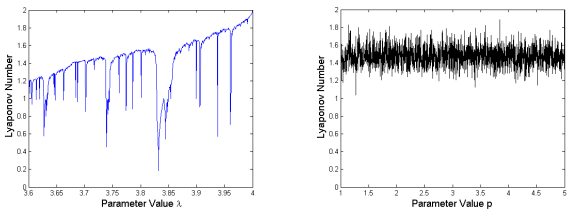
Following Kocarev et al., we propose that system designers require two attributes of a image-based cryptographic scheme: *diffusion* and *confusion* [2]. We define *diffusion* as the ability to spread highly correlated information of the input image around such that little, if any, statistically significant structure remains in the encrypted image. Without good diffusion, an intelligent attacker given an encrypted image could recover partial or complete information about the original plain-text image. Similarly, we define *confusion* as the scheme's ability to transform input images that differ only slightly (e.g. by only a few pixels) into radically different encrypted images. Without confusion, an attacker with the ability to provide chosen input images to the system could exploit these privileges to uncover the value of the secret keys. Within this work, we will focus on designing for diffusion goals, and later show how confusion vulnerabilities can be exploited.

Designing a Chaotic Cryptosystem: Applying mathematics to choose the best map — Although many well-known chaotic maps exist (e.g. circle map, Henon map, etc.) choosing one for a cryptosystem becomes a challenging mathematical design exercise. Several considerations must be balanced, including the diffusion performance of the map, the availability of a wide parameter space for selecting unique secret keys, and the map's behavioral consistency within the chaotic regime [2]. Each of these is discussed below with emphasis on litmus tests for making decisions between alternatives.

First, a map's diffusion performance must be carefully considered as part of the overall encryption process. An ideally diffusive map would transform any input to a uniform distribution across the possible pixel locations and values, so as to prevent any correlation with the input. Most chaotic maps operate within the real numbers, while digital images are usually composed of integers, so the algebraic transformation between these domains must be chosen to preserve a map's diffusive properties. Overall performance of a scheme can be measured via correlation statistical measurements between the original image and the cipher image, as shown in [5].

Second, the chosen map should offer a wide parameter space from which to select keys. Although chaotic maps are real-valued functions, in practice computation limits the precision at which keys can be specified due to storage and run-time restrictions. Because of this, a designer must be aware of these limitations and plan ahead to choose a map that will offer many more unique keys than will ever be necessary for the intended application.

Finally, behavioral consistency with respect to parameter space is perhaps most important when choosing a map. It is no good to have many keys available if only a few of them yield reliable mixing behavior. Kocarev et al. refer to this desired property as robust chaos, meaning that the chaotic properties of the map should be consistent and independent of initial conditions, parameter values, etc. within the chosen key space [2]. Another name for this idea of structurally uniform behavior independent of parametrization is *ergodicity*.



(a)Original Logistic Map (b)Revised Map, $\lambda = 4$

FIG. 1: Side-by-side comparison of Lyaponov number L for both the original logistic map and its proposed cryptographic modification. The threshold for chaotic behavior is $L \geq 1$. The original map contains several degraded, non-chaotic states throughout its parameter space, while the proposed map has consistent chaotic behavior.

Armed with these objectives, we now set out to choose a map. The one-dimensional logistic map shown below has a well-studied region of chaotic behavior for $\lambda \in (3.6, 4.0)$ and was shown to successfully meet diffusion objectives in [5].

$$x_{n+1} = \lambda x_n (1 - x_n) \tag{1}$$

However, it is worth noting that although chaotic behavior is common throughout this interval, it is not the only possibility. A plot of the Lyaponov number L (a common litmus test for a system's chaotic-ness) as a function of λ is shown in figure 1(a). The behavior descends below the $L = 1$ chaotic threshold at several points, indicating that the map is not structurally stable (ergodic) across the parameter space and thus should be used very cautiously in cryptographic applications to ensure a non-chaotic key is not chosen.

To overcome this disadvantage, Kocarev et al. propose a slightly modified logistic map [2]. They fix $\lambda = 4$, which is firmly in the chaotic regime, and add an additional parameter p which can be any real number. The revised governing equation becomes

$$x_{n+1} = 4(x_n + p)(1 - (x_n + p))(\ \texttt{mod}\ 1) \tag{2}$$

where the `mod` operator removes all but the fractional component of its input, so that $x_n \in (0, 1)$ for all n.

In contrast to the original map, this revised version exhibits stable chaotic behavior throughout a wide parameter space, as evidenced by the plot of the Lyaponov number in figure 1(b). We can see from the plot that this map has several desirable cryptographic properties: it falls consistently within the chaotic regime $L > 1$ and allows a key space at least four times as wide as the original logistic map. We will adopt this particular map in our sample encryption algorithm in the next section.

Sample Algorithm: Shuffle and Mask Encryption via the Logistic Map — Armed with the design considerations described above, we now propose a simple sample encryption algorithm that encrypts grayscale images

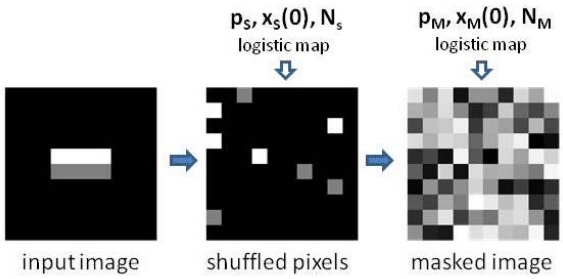


FIG. 2: Diagram of the two-step shuffle and mask scheme proposed. The shuffle process rearranges the input image pixels to diffuse correlation between pixels, while the mask phase scrambles pixel values to prevent statistical attacks or partial information. Each step requires a unique pair of keys: $p, x(0), N$ to generate a stream of pseudo-randomness from a revised logistic map.

based on the revised logistic map. We suggest that this scheme can yield results that have strong diffusion properties which make it resistant to the most popular statistical attack scenarios.

We begin with a computational definition of a grayscale image G as an M by N pixel matrix where the pixel value located in the ith row and jth column is given by $G(i, j)$. The total number of pixels is $P = M \cdot N$. We define each pixel to have a grayscale shading intensity value that ranges among the integers from 0 (black) to 255 (white). This definition allows each pixel to be represented by one byte (8 bits) of information, since $2^8 = 256$ possible values.

At the start of encryption, a grayscale input image G is provided to the encryption scheme along with a set of parameters and initial conditions that form the secret keys. The scheme produces a ciphered image C that is also a grayscale M by N matrix with pixel intensity within $[0, 255]$.

The basic process behind this scheme is a two-stage endeavor that alters pixel position and grayscale value to reduce the highly correlated properties of most images. The first step is a random shuffle, in which pixels are rearranged according to the iterative output of our revised logistic map with parameter p_S and initial condition $x_S(0)$. The second step is a masking process, in which another set of keys p_M and $x_M(0)$ into the revised map equations yields a stream of pseudo-randomness that we use to disguise the gray-level value of each shuffled pixel. An overall diagram of the system, including inputs and visualization of intermediate steps, is shown in figure **??**

First, we begin with the shuffling process. The intent of this step is to destroy any correlational information each pixel may share with its local neighborhood by scattering pixels throughout the image. This shuffling process enables diffuse mixing which (we hope) limits an attacker's ability to consume a ciphered image and backtrack to the original image via statistical guesswork.

The exact shuffle process we propose is as follows: first, iterate the revised logistic map using shuffle keys p_S and $x_S(0)$ for N_S transient iterations and then accumulate the next P values, one per pixel in the image G. We can label these accumulated values $x(1) \ldots x(P)$. Next, we sort this vector x in ascending order to form vector x_S. This sorted vector as the basis of the shuffling process. The transformation x to x_S can be described as an index reassignment vector i_S. For example, if $x(1)$ was the largest value of x, then since x_S is ranked in ascending order we would find $x_S(P) = x(1)$. We could then assign $i_S(1) = P$, which means the item in position 1 was moved to position P. Given this index reassignment vector i_S, rearrange the elements of G into a new shuffled image S such that $S(i_S(k)) = G(k)$.

Next, comes the mask process. The intent of this step is to prevent an attacker from using global information from the shuffled image to draw conclusions about the original. For example, even without the ability to reassemble a shuffled satellite photo a clever attacker could take a histogram of the intensity values and perhaps draw conclusions about the type of terrain being observed. A good cryptosystem should yield absolutely no information about the secret image, and so a masking process is needed. The tricky part of design is to create a mask that can be inverted with the secret key to recover the original value.

To begin, we apply masking keys p_M and $x_M(0)$ to the revised logistic map through N_M transient iterations and then accumulate P values into a vector x_M. Next, the produced vector x_M should be scaled and discretized from its continuous range $x_M(i) \in [0, 1]$ to a discrete integer value $Z_M(i)$ within $[0, 255]$. Mathematically, $Z_M = \texttt{floor}(256 \cdot x_M)$. Now within Z_M we have a unique byte for each pixel intensity value, and we can combine these via the bitwise exclusive-or (`bitxor`) operation to produce a masked pixel intensity.

We choose the function `bitxor` because of its simple invertible properties, since our masking process must be undone in decryption. Mathematically, `bitxor( bitxor ( x ) )` $= x$ for all values of x, which means recovering the shuffled image S from the cipher image $C = \texttt{bitxor}(S, Z_M)$ is just a matter of reproducing the vector Z_M from the iterated map given the secret keys and performing the operation $S = \texttt{bitxor}(C, Z_M)$. We can complete the decryption process by reproducing the index reassignment vector i_S and computing $G(k) = S(i_S(k))$. This completes the algorithmic description. In the next section, we will study the advantages of this scheme against possible attack scenarios.

Security Analysis: Diffusion and Confusion — We suggested in our first section that a good cryptosystem should have diffusion and confusion properties. How does our simple cryptosystem fare? Let us first examine diffusion properties.

Diffusion has two goals. First, we hope to avoid statistical correlation from the original image G being evident in the cipher image C. Second, we wish to avoid slightly different keys producing similar map dynamics

that would yield any information about the original image. We can test our results with respect to both goals by considering the sample image and its encryption provided in figure 3.

Although a sophisticated correlation assessment is beyond the scope of this paper, we can offer several results supporting the diffusive properties of the scheme. First, we show the process has acceptable mixing properties. Observe that the grayscale histograms within figure 3 indicate that the encrypted image's distribution of grayscale intensities is near-uniform. This uniformity prevents an attacker from recovering partial information about the hidden image. Second, we show that the keys within this map are extraordinarily sensitive. After implementing the scheme in MATLAB, we attempted to unlock images through a decryption process that used near-miss keys. We found that if secret key was p, x_0, even p + `eps` and x_0 + `eps` (where `eps` is the smallest quantity computable in MATLAB) produced a senseless, noisy decrypted image with an underlying uniform histogram. Further tests are certainly necessary to formally verify the scheme's diffusive properties, but we believe the presented scheme holds promise in this design objective.

Confusion is a more problematic design goal within image encryption. Ideally, a strong confusing scheme would take two similar inputs and produce radically different outputs. Unfortunately, the scheme we outlined above does not meet this requirement and is in fact completely breakable because of this. The type of attack necessary is known as a chosen plaintext attack. If the attacker has access to the an encrypting "black box" implementing this scheme, she can use specially chosen inputs and some clever comparison to completely recover both the masking sequence Z_M and the index shuffling vector i_S, which are sufficient for encrypting or decrypting any image within the scheme. This simple attack, suggested by [6], thus completely breaks the shuffle and mask scheme.

The attack begins by breaking the mask phase. The crux of this step is the choice of a unique input image U_Z such that all pixels are uniformly zero. This uniform input renders the shuffling process worthless, since $\texttt{shuffle}(U_Z) = U_Z$. The fact that all pixels are zero then allows the recovery of each mask byte $Z_M(i)$, as $\texttt{bitxor}(Z_M(i), 0) = Z_M(i)$ for all i. Recovering the secret keys for this phase is unnecessary, as the Z_M result is all that is necessary for encrypting or decrypting the mask phase.

With the ability to break the mask, the attacker can then dissect the shuffle phase of encryption. First, the attacker chooses some grayscale image G_Z in which the first 255 pixels have values $1, 2, \ldots 255$ and the rest of the pixels are uniformly zero. Next, the attacker feeds this input G_Z into the scheme, receives the encrypted image R_Z, and then recovers the shuffled intermediate image $S_Z = \texttt{bitxor}(R_Z, M)$. Using S_Z, the attacker finds the first 255 values of the index reassignment vector i_S by determining where the pixels at positions 1 thru 255 were scattered. Since each non-zero pixel value is unique, discovering this reassignment should be a simple search through the shuffled image S_Z. This entire process of constructing an image with 255 consecutive pixels differing from U_Z, encrypting it, and then uncovering the reassignment for those 255 indices is then repeated for each batch of 255 consecutive pixels within G_Z until the complete vector i_S has been recovered. This completes the break of the shuffle and mask scheme. Of course, this attack is not trivial and requires input/output access to an encrypting "black box." However, the attack is both possible and computationally feasible, so it should be a concern to system designers.

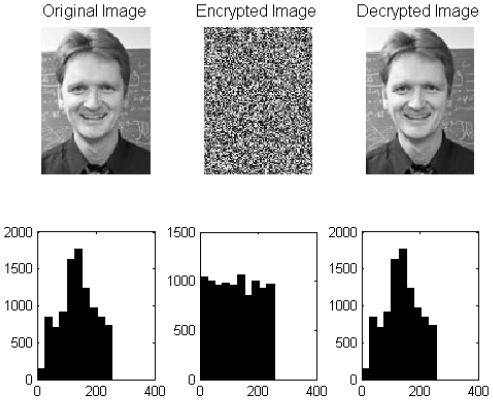


FIG. 3: Example shuffle and mask encryption and decryption applied to gray-scale image with parameters $p_S = 3, x_S(0) = 0.1, p_M = 4$ and $x_M(0) = 0.1$ driven through $N = 100$ transient iterations. Histograms of image pixel values are shown for each step.

Conclusion — Chaotic maps provide a promising alternative to conventional image encryption algorithms, with advantages of run-time efficiency, scalability, and applicability in embedded hardware applications. However, system designers should actively consider both confusion and diffusion aspects of a chaos-based encryption scheme, as each is crucial to producing unassailable image confidentiality.

For our example scheme, confusion proved to be a weakness, and this is typical of other chaos-based encryption proposals [3]. Unfortunately, this objective is often missing entirely from validation (e.g. in [5]). Avoiding mask recovery and differential attacks based on highly-similar input images should be a primary goal of future work. We suggest perhaps several layers of alternate masking and shuffling might provide more reliable confusion. A more general goal should be to ensure that any mask used cannot be recovered through the choice of a degenerate input (e.g. U_Z). These challenges and others will ensure that image encryption will remain widely-studied in coming years.

[1] F. Belkhouche, U. Qidwai, I. Gokcen, and D. Joachim. "Binary Image Transformation Using Two-Dimensional Chaotic Maps." Proc. International Conf. on Pattern Recognition **4** (2004).

[2] L. Kocarev and G. Jakimoski. "Logistic map as a block encryption algorithm." Physics Letters A **289**, pp. 199206. (2001).

[3] G. Chen , Y. Mao , and C. K. Chui. "A symmetric image encryption scheme based on 3D chaotic cat maps." Chaos, Solitons and Fractals **21** pp 749-761. (2004).

[4] M. Usama, M. K. Khana, K. Alghathbara, and C. Leeb. "Chaos-based secure satellite imagery cryptosystem." Computers & Mathematics with Applications **in press** (2010).

[5] N.K. Pareek, Vinod Patidar, K.K. Sud. "Image encryption using chaotic logistic map." Image and Vision Computing **24** pp. 926934. (2006).

[6] Cahit okala and E. Solak. "Cryptanalysis of a chaos-based image encryption algorithm." Physics Letters **373** 15 pp. 1357-1360. (2009).