Steps towards improving the security of chaotic encryption

Blair Fraser,* Pei Yu,† and Turab Lookman‡
Department of Applied Mathematics, University of Western Ontario, London, Ontario, Canada N6A 5B7
(Received 23 January 2002; published 19 July 2002)

We present a method in which a chaotic signal is used to mask a message securely. It depends on separating the two tasks of synchronizing the chaotic oscillators and encrypting the message. A sporadic drive together with a function f of the ciphertext and response system variables is used to make extraction difficult. We give a choice of f that makes the method similar to a one-time pad, with pseudorandom numbers provided by the chaos.

DOI: 10.1103/PhysRevE.66.017202 PACS number(s): 05.45.−a, 81.30.Kf, 64.70.Kb, 61.72.Dd

I. INTRODUCTION

It is well known that two identical low dimensional systems exhibiting chaotic motion may be synchronized by giving one system partial information about the state of the other. Since this chaos is unpredictable, the complete state of the chaotic oscillator is difficult to predict or reproduce without an identical nonlinear system. This makes chaos a good candidate for masking a message. Pecora and Carrol [1] noticed that, by adding a message with a small amplitude to the chaos, the signal is still close enough to the exact chaotic mask to allow the dynamics of the two systems to synchronize, if they share all the same parameters in the equations—which become the private keys to the method. It is assumed in this method that the nonlinear system in use is known by an intruder, as is the function that combines the chaotic mask with the message (addition in this case). This must be retained in any modifications to the method. By the use of such synchronization, we can reproduce the mask at the receiving end. Simply subtracting this reproduced mask from the received encrypted message leaves the original plaintext message. We note that many other popular methods of using chaos to encrypt a message exist [2,4], including shift keying, which are not addressed in this paper.

There has been much interest in using synchronized low dimensional chaos for data encryption. The encryption method is reasonably fast, simple to implement, and has been assumed secure. However, several security flaws have recently become apparent that allow the underlying message to be revealed without a knowledge of the private keys or even of the nonlinear system itself in some cases. Increasing the dimension of the chaos does not appear to be the way to improve security [3].

The addition of the message to chaos changes the return map of the chaotic signal, as may be seen by plotting maxima and minima of the masked message. If the return map is close to being one dimensional, then the message can be detected as deviations to this quasi-one-dimensionality. This is the case in the return map of the Lorenz equations as used in [1], as well as many other equations. For a discussion of how to extract the plaintext message without keys, see [5]. The difficulty can be repaired adequately as shown in [6], by discretizing the signal and the message, thus blurring the return map, and preventing the message from being easily decrypted with simple signal processing techniques. This method improves security even when using systems that exhibit a low dimensional chaotic trajectory, as an intruder sees only discrete points on the attractor. The aim of this work is to propose methods that can encrypt messages more securely by destroying more information about the message that may have otherwise been gathered from the cyphertext. We consider a digital implementation where the systems are solved numerically. The chaos is used to create a pseudorandom bit stream and this bit stream is used as a mask for the message. After introducing the notation in Sec. II, we discuss in Sec. III the desirable elements of a secure method and show how the proposed method overcomes flaws in previous approaches. Transmission errors are considered in Sec. IV and we highlight possible sources of weakness in Sec. V. In Sec. VI we briefly discuss how this method may be used in an analog system, and some practical hurdles in this implementation.

II. NOTATION

For ease of discussion, we assume that the system has three variables. The proposed method is general enough to use on any synchronizable chaotic system, continuous or discrete; however, in this paper we will often use the Lorenz system as an example as it is the system that appears most in the study of encryption with chaotic ordinary differential equations. We write the generic system as

$$\dot{x}=g_1(x,y,z), \quad \dot{y}=g_2(x,y,z), \quad \dot{z}=g_3(x,y,z).$$

The drive system will be denoted $\mathbf{d}$, and the variables of the drive system will be denoted $\mathbf{d}_x$, $\mathbf{d}_y$, and $\mathbf{d}_z$. Similarly, the response system will be called $\mathbf{r}$, and its independent variables $\mathbf{r}_x$, $\mathbf{r}_y$, and $\mathbf{r}_z$. We assume that the x variable is being used as a drive, i.e., $\mathbf{r}_x$ is being driven by $\mathbf{d}_x$. The plaintext message will be denoted by m, and the ciphertext message by c.

III. REPAIRING THE FLAWS

The security holes in chaotic encryption rely on how simply the chaos and message are combined to produce the ci-

*Electronic address: bfraser@julian.uwo.ca
†Electronic address: pyu@pyu1.apmaths.uwo.ca
‡Electronic address: txl@viking.lanl.gov

phertext. Conventionally, the message m is reduced in amplitude to be much quieter than the chaotic mask $\mathbf{d}_x$; the two are then superimposed to give c. We are forced to use such a simple function of the mask and message because we need the combination m and $\mathbf{d}_x$ to look very similar to $\mathbf{d}_x$ so that it can be used to synchronize the response system with the drive system. We could imagine a much more complicated function of m and $\mathbf{d}_x$, denoted by $f(m,\mathbf{d}_x)$, that looks like neither the message nor the mask, and would be much more complicated to cryptanalyze. All useful information about the message contained in the return map could be completely scrambled by a sufficiently complex combination. In fact, we could even imagine combining the message with more than one mask from the transmitter, supplied by more than one variable of the nonlinear system. For example, in our system of three dynamical variables we could use a function of all of the three variables and the message $f(m,\mathbf{d}_x,\mathbf{d}_y,\mathbf{d}_z)$ to create a complicated ciphertext message. The problem is then using this as a drive. We mention again that both f and the nonlinear system are assumed to be known to an intruder. The keys to the method will be parameters in the differential equations, analogous to the parameters σ, r, and b used in [1], and only these keys are unknown to an intruder.

The information contained in the signal that will be used for the drive is potentially useful to an intruder. However, scrambling this information to make it more difficult for an intruder to use also makes it impossible for the response system to use as a drive signal, destroying any hope of synchronizing the oscillators.

To allow the use of this complicated function of the variables and the message, we must appreciate that there are two tasks here. One is to synchronize the two oscillators to create identical masks, in a robust way, and the other is to use masks created by the chaos to encode and transmit data. Separating these tasks will allow us to use the chaotic masks in any way we desire, and still synchronize the two systems. These changes to the method allow us increase security. Based on the results in [6], we can use a sporadic drive [7] to accomplish this separation of tasks. We consider no message here, just the synchronization of two identical oscillators. We can synchronize two oscillators if we send a short drive pulse every T_n time units, so long as T_n is shorter than some maximum time T_H. The time T_H will be determined by the choice of oscillator used. In the Lorenz system, T_H was found to be ≈ 0.31 if the x variable is used as a drive signal [6]. During the time the drive is off, nonzero errors will grow until the next drive pulse. When the method is implemented digitally, once the chaos is synchronized to machine precision, only bit errors will cause the oscillators to come out of synchronization.

We now consider the creation of the transmitted message as a sequence of packets, where a packet contains a header, and some encrypted data. The header contains a single piece of the sporadic drive signal. In our example, the header will be $\mathbf{d}_x$ at regularly spaced intervals T_n. The header is fed straight into the drive of the response system to keep the two oscillators synchronized. After the header, the encrypted data are sent. They are not fed as a drive to the response system, but instead are sent to the inverse of f, along with $\mathbf{r}_x$, $\mathbf{r}_y$, and $\mathbf{r}_z$. If the two oscillators are synchronized then m can be recovered through $f^{-1}(c,\mathbf{r}_x,\mathbf{r}_y,\mathbf{r}_z)$. Following this is another packet, containing a header, and some encrypted data, which is dealt with in the same manner.

As the synchronization of the oscillators and the encryption of the message are separated, we have complete freedom in how we choose f, as long as f is invertible. We can choose f in such a way that small deviations in the keys become more sensitive. In conventional chaotic encryption, keys that are not identical but are very close can still be used to synchronize the two systems very well, thus reducing the keyspace of the encryption. We will not alter this; close keys will still synchronize the systems closely. Instead, we will construct f to be very sensitive to small changes in the chaotic inputs $\mathbf{d}_x$, $\mathbf{d}_y$, and $\mathbf{d}_z$ so that close synchronization will not be good enough to decrypt the message. There are many ways to create such an f. The point of this encryption is that we have complete freedom in choosing f. Here we will discuss one such f that can be used to give arbitrarily sensitive keys.

The way we construct f will be to take the least significant digits of $\mathbf{d}_x$, $\mathbf{d}_y$, and $\mathbf{d}_z$ to create three bit masks. The way we have chosen to do this is with the function

$$\text{mask}_x = \lfloor (s\mathbf{d}_x - \lfloor s\mathbf{d}_x \rfloor)(2*\text{INT\_MAX}) \rfloor. \tag{1}$$

Similar functions using $\mathbf{d}_y$ and $\mathbf{d}_z$ create mask_y and mask_z. Here s is a number used to control how sensitive the function is to the chaotic inputs, and 2*INT_MAX is the maximum unsigned integer that can be represented by the computer. This results in three integer sized bit masks. For example, if $\mathbf{d}_x = 27.183\,948\,475\,701$, we are taking the least significant digits of $\mathbf{d}_x$. If $s = 10^6$, we take 475 701 and create a bit mask out of this. It is the most significant digits of the chaos that are easily predictable. We dispose of these most significant digits, and use the digits that are sensitive when close but not exact keys are used. These bits are also sensitive to numerical error, which will be dealt with later. Once mask_x, mask_y, and mask_z are created, a XOR operation is performed with the message to create the ciphertext

$$c = m \oplus \text{mask}_x \oplus \text{mask}_y \oplus \text{mask}_z. \tag{2}$$

If the receiver can reproduce mask_x, mask_y, and mask_z from $\mathbf{r}_x$, $\mathbf{r}_y$, and $\mathbf{r}_z$, then the message may be recovered by a XOR operation on these masks with the received ciphertext c,

$$m = c \oplus \text{mask}_x \oplus \text{mask}_y \oplus \text{mask}_z. \tag{3}$$

Notice that an intruder must be able to reproduce $\mathbf{d}_x$, $\mathbf{d}_y$, and $\mathbf{d}_z$ down to the sensitivity determined by s to be able to read the message. This differs from conventional chaos encryption, which only requires that the intruder work with $\mathbf{d}_x$ from a ciphertext signal that looks *almost* like $\mathbf{d}_x$. In our method, even if the intruder could reproduce $\mathbf{d}_x$ exactly, the message could not be decrypted without knowing $\mathbf{d}_y$ and $\mathbf{d}_z$ as well. We show the results of a correct decryption, and an attempted decryption with close but not exact keys (within 2%), in Fig. 1. We emphasize that this choice of f is only an example. The choice of this function is completely arbitrary.

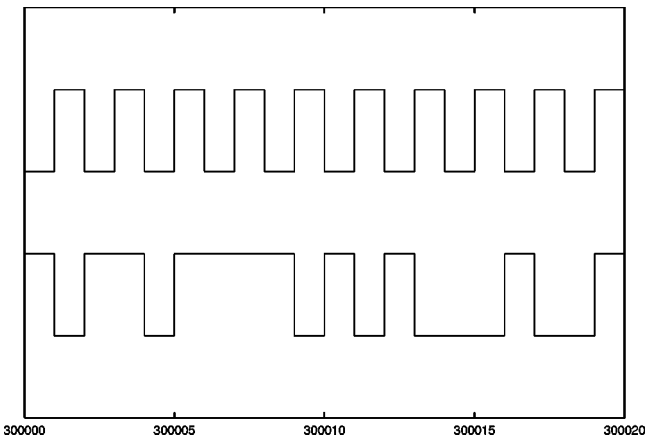

FIG. 1. Decryption of an oscillating bit stream with exact keys (top), and close but not exact keys (bottom).

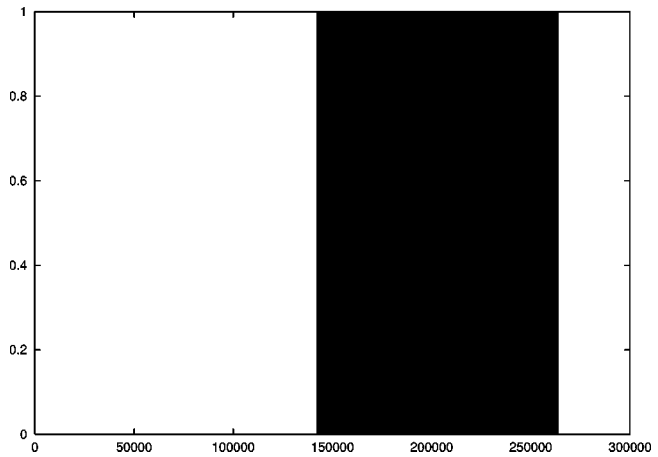

FIG. 2. Error in each bit of recovered message when a bit error occurs in header at bit 142400.

We have used this function because of its ease of use. It is not the best choice, because the first bits in the mask are less secure than the later bits since they come from more significant digits in the chaotic variables. We could do better by subjecting this mask to a hash function to make the security across bits uniform.

This particular choice of f makes the method look very similar to a one-time pad, with pseudorandom numbers provided by the chaos. A one-time pad performs a XOR operation on a message with *real* random numbers, and is provably secure. The security of a one-time pad relies on the mask being truly random, and the pad being shared by *only* the transmitter and receiver. The mask in our method is not truly random, but if it displays the properties of random numbers well enough and is very hard to reproduce with the information freely available, then the plaintext will be secure to any reasonable amount of computing power. Since the pad is reproduced from synchronization data, the pad does not need to be shared by the transmitter and receiver before the secure communication is to occur. Note that the actual pad created by the transmitter depends on the random initial conditions chosen for the oscillator, which should be different every time to ensure the pad is only used once. The receiver does not need to know these random initial conditions, for when the communication is initialized several ''handshake'' headers are sent to synchronize both systems before any data are sent. During this handshake, even close initial conditions will diverge, ensuring the uniqueness of every pad by any reasonable standards. (The pad will not be unique since the computer can only represent a finite number of initial conditions; however, this finite number is large enough that an intruder should never see a pad repeated.)

IV. ROBUSTNESS TO BIT ERRORS

We consider burst noise that affects one or more bits in a packet, either in the data or in the header. If there are errors in the encrypted data of a packet, then this packet will be destroyed. The masks will be reproduced correctly at the receiver's end, but if the input ciphertext contains errors, then the output of f^{-1} will also contain errors. Since the oscillators are still synchronized, the next packet will not be destroyed by this bit error. A far more serious error is an error in the packet header. If the response system is fed false information as a drive, then the two oscillators fall out of synchronization. The masks cannot be reproduced from the unsynchronized response system, and the packet is destroyed. In addition to this, all packets are destroyed until the two oscillators come back into synchrony. We can see this in Fig. 2, where the Lorenz system is used; approximately 14 kilobytes are lost before the oscillators are again synchronized. This length of time depends on both the oscillator used and the value of s. If we set s such that we demand the oscillators be in synchrony right down to machine precision, then the recovery time is longer. This is a trade-off against the sensitivity of the keys. Demanding the masks match exactly down to the last bit makes the keys incredibly sensitive and thus increases the available keyspace. However, it also increases the recovery time in the event of an error in a header. But, in any case, only a finite number of packets are destroyed by any errors in transmission.

V. ADVANTAGES, DRAWBACKS, AND POTENTIAL FLAWS

Encrypting a message in this way is simply using a pseudorandom number generator to create a one-time pad. The security of this type of encryption lies wholly in the random number generator. We have changed how the random numbers are generated. By doing so, we gain an important advantage. In many pseudorandom number generators, if synchronization is lost, then it can never be regained in a secure manner. Each random number depends on a small number of previous numbers only, so the first random numbers, the seeds, determine all succeeding random numbers. If synchronization is lost, one cannot simply have the transmitter send all the state information of its random number generator so that the receiver can resynchronize. This would allow an intruder to synchronize as well. Thus all bits of the message after a synchronization error are lost. By using synchronized chaos, we send some resynchronization information to the response system in each header that allows it to resynchro-

nize after loosing a finite number of bits. It is a much more difficult task for an intruder to use this synchronization information. The intruder must reproduce $\mathbf{d}_x$, $\mathbf{d}_y$, and $\mathbf{d}_z$ exactly from discrete pulses of $\mathbf{d}_x$, or, alternatively, reproduce the private keys from this same information. The chaotic equations are not fixed in this method, so the intruder must find a general way to do this, one that will work on all synchronizable low dimensional chaotic systems, continuous or discrete.

One drawback to this method is that the sensitivity to keys also creates sensitivity to numerical error. The systems are chaotic, so large numerical errors will occur, but if we can be assured that exactly the same numerical error will occur at both ends, there is no problem. Unfortunately, this is not always so easy to ensure. Machines that have different precisions for floating point numbers will not decrypt if s is set to be very sensitive. This amounts to standardizing how the floating point numbers are used in the method. This may be done by using a language that determines floating point numbers across platforms, such as JAVA, or by standardizing the hardware that executes the encryption. In addition, the same numerical algorithms must be used at both ends, but this just amounts to standardizing the software, and does not pose a problem. We should also note that the headers increase the size of the message. This increase will vary, depending on the nonlinear system used, by its value of T_H. Obviously a large T_H allows a lot of encrypted data to be sent after a synchronization header and will still be able to resynchronize after an error. The Lorenz system allows transmission of approximately 36 bytes of data after 8 bytes of synchronization information.

VI. CONCLUSION

We presented a private key method of encryption using synchronized chaos. We assumed that the nonlinear system and the function f are known to an intruder. The scalar parameters of the nonlinear equations, in the case of the Lorenz system r, b, and σ, remain unknown to the intruder and become the private keys to the method. The return map attacks against the conventional method are repaired, and we believe as few as possible new potential attacks are opened. This method is part chaos encryption and part pseudorandom number one-time pad encryption. We have kept the strongest advantages of both methods, while losing as little as possible in the combination of the two. The security of this method is not a closed matter. The easiest attacks to see are those attacking the random numbers, or gathering information contained in the headers and trying to reproduce either the keys or the mask. However, discrete points on the trajectory of a chaotic attractor have been used in [6] and found to be secure against the simple signal processing techniques used to break chaotic encryption as in [5]. Further, we have shown through simple tests of randomness, including Knuth's spectral test, and a test of the dimensionality of the structure of the random numbers, that the random numbers will not fall victim to the simplest of attacks. However, this analysis is insufficient to state that the random numbers are *cryptographically* secure. We do not address if enough information can be obtained from the headers to attack the message, but we do not see any simple ways to achieve this. Before this method is implemented and used for secure communication, both of these points *must* be analyzed in a more comprehensive and in depth study [8]. Finally, we mention that this method may also be applied to analog communications by creating f to be a complex analog function. However, since the receiver must know when it is receiving a header and when it is receiving encrypted data, there must be synchronized switching between the transmitter and receiver. This is trivial in a digital system, but not as trivial in an analog implementation. It is, however, a solved problem used in all computer networks.

[1] Louis M. Pecora and Thomas L. Carrol, Phys. Rev. Lett. **64**, 821 (1990).
[2] M. S. Baptista, Phys. Lett. A **240**, 50 (1998).
[3] K. M. Short and A. T. Parker, Phys. Rev. E **58**, 1159 (1998).
[4] Tao Yang, Lin-Bao Yang, and Chun-Mei Yang, Physica D **124**, 248 (1998).
[5] Gabriel Pérez and Hilda A. Cerdeira, Phys. Rev. Lett. **74**, 1970 (1995).
[6] Y. Y. Chen, Europhys. Lett. **34**, 245 (1996).
[7] Toni Stojanovski, Ljupco Kocarev, and Ulrich Parlitz, Phys. Rev. E **54**, 2128 (1996).
[8] Pei Yu, Blair Fraser and Turab Lookman (unpublished).