Using Chaos for Data Encryption and Secure Communication

MATH 467/767

Eric Sund, Ari Blondal, James Andrews

***Abstract*— Chaotic systems are deterministic systems which show unpredictable, yet long-term behaviour, that have highly sensitive dependence to initial conditions and/or parameters. While this unpredictability seems very useful in in the encoding of signals and communications, in practice it is difficult because of the sensitivity of the system to any variations or errors.**

We discuss a solution proposed by Cuomo et al. (1993) [2], in which a transmitting system and receiving system are described, with the transmitting system driving the receiving system such that they automatically synchronize. We then discuss the use of this system in signal encoding, showing some practical examples.

I. Lorenz Equations and their Chaos

The Lorenz equations are given by

$$\begin{aligned}\dot{x} &= \sigma(y-x)\\ \dot{y} &= rx - y - xz\\ \dot{z} &= xy - bz\end{aligned} \tag{1}$$

for parameters $\sigma, r, b > 0$. Originally derived to model convection rolls in the atmosphere, the Lorenz equations can also model lasers, dynamos, and much more.

The Lorenz equations exhibit chaotic behavior, defined as bounded aperiodic long-term behaviour with sensitivity to initial conditions, at certain parameter values. We choose $\sigma = 16$, $r = 45.6$, $b = 4$ in order to induce chaos in the system. With these parameter values, the system has a positive Lyapunov exponent, indicating the rate of separation between trajectories is exponential. Therefore the system is chaotic. In fact, $\lambda \approx 1.44$ in this case (Fig 1).

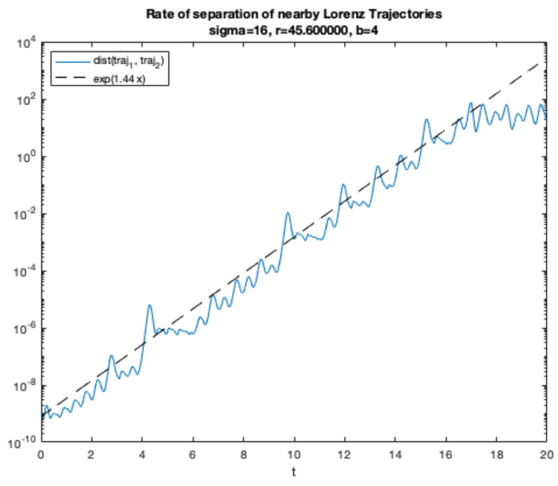

Fig. 1: A Lyapunov exponent calculation for an example trajectory.

Figure 1 plots time against the distance between trajectories. The plot then fits a straight line to approximate the slope as best as possible. $\sigma = 16$, $r = 45.6$, $b = 4$. Notice that all values reach an eventual equilibrium as trajectories reach a strange attractor, at which point seperation remains constant.

Figure 2 shows the numerical results after 1500 iterations of a Lyapunov exponent calculation for all three dimensions. Simulating the system with a Forward Euler method, We observe the Lyapunov exponent appears to converge to about 1.48, close to our sample curve's exponent.

We can give some analytical intuition as to why the Lyapunov exponent is positive. $D\vec{F}^{(n)}$ is the linearization of the rate of change of the nth iteration of a 3D Lorenz Map, where D is the Jacobian operator and $\vec{F}$ is the function defining the 3D Lorenz Map. The Jacobian is:

$$D\vec{F} = \begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} & \frac{\partial f_1}{\partial z} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} & \frac{\partial f_2}{\partial z} \\ \frac{\partial f_3}{\partial x} & \frac{\partial f_3}{\partial y} & \frac{\partial f_3}{\partial z} \end{bmatrix} = \begin{bmatrix} -\sigma & \sigma & 0 \\ -z+r & -1 & -x \\ y & z & -b \end{bmatrix} \tag{2}$$

where

$$\langle f_1, f_2, f_3 \rangle = \langle \dot{x}, \dot{y}, \dot{z} \rangle$$

As defined in (1).

If we let $H = D\vec{F}$, then HH^T has one positive and two negative eigenvalues. The Lyapunov exponents are expressed as:

$$\lambda_i = \lim_{t\to\infty} \frac{\ln(a_i(t))}{2t}$$

where λ is the ith Lyapunov exponent and a_i is the ith eigenvalue of HH^T. Since we have one positive eigenvalue, the maximal λ_i is positive, and there is chaos.

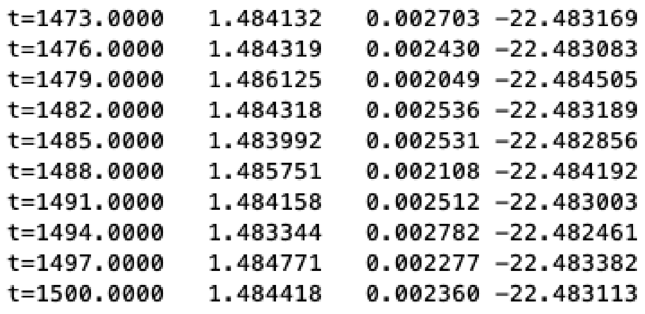
```
t=1473.0000    1.484132    0.002703 -22.483169
t=1476.0000    1.484319    0.002430 -22.483083
t=1479.0000    1.486125    0.002049 -22.484505
t=1482.0000    1.484318    0.002536 -22.483189
t=1485.0000    1.483992    0.002531 -22.482856
t=1488.0000    1.485751    0.002108 -22.484192
t=1491.0000    1.484158    0.002512 -22.483003
t=1494.0000    1.483344    0.002782 -22.482461
t=1497.0000    1.484771    0.002277 -22.483382
t=1500.0000    1.484418    0.002360 -22.483113
```

Fig. 2: The numerical results corresponding to running a Lyapunov spectrum calculation for 1500 iterations using a forward Euler simulation of the Lorenz system. The first column indicates iterations, while the other three columns represent the three dimensional Lyapunov exponents.

II. Synchronizing Chaos

In order to use chaotic systems to encrypt or encode signals, we need to determine a way of recreating a specific trajectory, at least within certain error bounds. We wish to synchronize the chaos in these two systems. Cuomo [2] notes that a chaotic system is self-synchronizing if it can be decomposed into two or more subsystems. Coincidentally, the system from (1) can be decomposed into:

$$\begin{aligned} \dot{x}_1 &= \sigma(y - x_1) \\ \dot{z}_1 &= x_1 y - b z_1 \end{aligned} \tag{3}$$

$$\begin{aligned} \dot{y}_2 &= rx - y_2 - x z_2 \\ \dot{z}_2 &= x y_2 - b z_1 \end{aligned} \tag{4}$$

It turns out that the difference between the trajectories x_1, x and z_1, z approach 0 as time tends to infinity. This means systems (3) and (4) can be used to re-create system (1).

Let J_1 be the Jacobian matrix of (3), and let J_2 be the Jacobian of (4), then:

$$\begin{aligned} J_1 &= \begin{bmatrix} \frac{\partial \dot{x}_1}{\partial x_1} & \frac{\partial \dot{x}_1}{\partial z_1} \\ \frac{\partial \dot{z}_1}{\partial x_1} & \frac{\partial \dot{z}_1}{\partial z_1} \end{bmatrix} \\ &= \begin{bmatrix} -\sigma & 0 \\ y & -b \end{bmatrix} \\ \lambda_{1,2} &= -b, -\sigma \end{aligned} \tag{5}$$

For J_2, Cuomo numerically derives the Lyapunov exponent, showing that it is negative [2]. As such,

$$\begin{aligned} \lim_{t\to\infty} |x_1(t) - x(t)| &= 0 \\ \lim_{t\to\infty} |z_1(t) - z(t)| &= 0 \\ \lim_{t\to\infty} |y_2(t) - y(t)| &= 0 \\ \lim_{t\to\infty} |z_2(t) - z(t)| &= 0 \end{aligned}$$

When the input to system (4) is $x(t)$, then the output $y_2(t)$ can be fed into system (3) to recreate a trajectory.

III. Constructing a Circuit

Cuomo constructed a transmitter circuit and a receiver circuit in his original paper. These were used to synchronize a waveform between two parties. The transmitter component of the circuit was scaled down to account for power supply limits. A convenient choice of scaling is

$$u = \frac{x}{10}, v = \frac{y}{10}, w = \frac{z}{20},$$

producing the new system:

$$\begin{aligned} \dot{u} &= \sigma(v - u) \\ \dot{v} &= ru - v - 20 \cdot uw \\ \dot{w} &= 5 \cdot uv - bw \end{aligned} \tag{6}$$

and its associated receiver circuit:

$$\begin{aligned} \dot{u}_r &= \sigma(v_r - u_r) \\ \dot{v}_r &= ru - v_r - 20 \cdot uw_r \\ \dot{w}_r &= 5 \cdot uv_r - bw_r \end{aligned} \tag{7}$$

It's worth noting that these two components are nearly the same, except that $u(t)$ lies within $\dot{v}_r$ and $\dot{w}_r$, which functions as the input to the receiver circuit. System (6)'s $u(t)$ waveform is transmitted and fed into system (7). (7) then synchronizes with system (6), so long as the parameters are identical. Why is this so? We'll show these two systems can synchronize perfectly.

Theorem 3.1: We define synchronization of two systems to occur when the difference of two systems approaches zero over time. Let

$$\begin{aligned} \vec{d}(t) &= \langle u(t), v(t), w(t) \rangle, \\ \vec{r}(t) &= \langle u_r(t), v_r(t), w_r(t) \rangle, \end{aligned}$$

as defined by equations (6) and (7). Then

$$\begin{aligned} \vec{e}(t) &= |\vec{d}(t) - \vec{r}(t)| \text{ and} \\ \lim_{t\to\infty} \vec{e}(t) &= 0. \end{aligned}$$

Proof: Let σ, r, b hold the same values in both (6) and (7). We denote the difference between trajectories in both systems as:

$$\begin{aligned} \dot{e}_1 &= u(t) - u_r(t) \\ \dot{e}_2 &= v(t) - v_r(t) \\ \dot{e}_3 &= w(t) - w_r(t) \end{aligned}$$

Plugging in (6) and (7) results in the following system:

$$\begin{aligned} \dot{e}_1 &= \sigma(u - v) - \sigma(u_r - v_r) \\ \dot{e}_2 &= (ru - v - 20uw) - (ru - v_r - 20uw_r) \\ \dot{e}_3 &= (5uv - bw) - (5uv_r - bw_r) \\ \dot{e}_1 &= \sigma(e_2 - e_1) \\ \dot{e}_2 &= -e_2 - 20u(t)e_3 \\ \dot{e}_3 &= 5u(t)e_2 - be_3 \end{aligned} \tag{8}$$

To show that system (8) has a stable, attracting fixed point at $(0, 0, 0)$, provided that $\sigma, b > 0$, we use the following Lyapunov function:

$$E(e_1, e_2, e_3) = \frac{1}{2}\left(\frac{e_1^2}{\sigma} + e_2^2 + 4e_3^2\right) \geq 0$$

Differentiating, we obtain:

$$\begin{aligned} \dot{E}(e_1, e_2, e_3) &= \frac{1}{\sigma} e_1 \dot{e}_1 + e_2 \dot{e}_2 + 4e_3 \dot{e}_3 \\ &= \frac{1}{\sigma} e_1(\sigma(e_2 - e_1)) + e_2(-e_2 - 20ue_3) \\ &+ 4e_3(5ue_2 - be_3) \\ &= e_1 e_2 - e_1^2 - e_2^2 - 20ue_2e_3 + 20ue_2e_3 - 4be_3^2 \\ &= e_1 e_2 - e_1^2 - e_2^2 - 4be_3^2 \\ &= \left(-e_1^2 + e_1 e_2 - \frac{1}{4}e_2^2\right) - \frac{3}{4}e_2^2 - 4be_3^2 \\ &= -\left(e_1 - \frac{1}{2}e_2^2\right)^2 - \frac{3}{4}e_2^2 - 4be_3^2 \\ &< 0 \end{aligned}$$

Therefore, $\lim_{t\to\infty} \vec{e}(t) = 0$, which shows the two systems (6) and (7) synchronize. ∎

IV. Simulation of circuit using MATLAB

In Cuomo's original paper, he illustrates chaotic behavior of the transmitter circuit using an analog-to-digital recording system and plots the solutions. We can actually verify this without needing such hardware. Here we wrote Matlab code to simulate the transmitter for a finite time period, then plotted the solutions in the uv-plane and uw-plane to get an idea of what the trajectories looked like. We used $u(t)$ from the transmitter as the driver, which was fed it into the receiver. Again, we've plotted the solutions, and notice they are the same as the transmitter's. We used Forward Euler's Method to numerically compute these solutions. As we can see, the plots have a chaotic strange attractor. We used the same parameter values as Cuomo's, that is, $\sigma = 16$, $r = 45.6$, $b = 4$. The initial conditions used are $u_0 = 1$, $v_0 = 1$, $w_0 = 1.05$. This shows how the two chaotic systems are able to be successfully synchronized. In figure 3, the transmitter circuit is plotted on the top row, and the receiver circuit is plotted on the bottom row. Even with all sorts of variations in the initial state of the receiver, it synchronizes almost instantaneously, and the graph produced resembles the linear relationship $u_r(t) \approx u(t), v_r(t) \approx v(t)$ in the long run.

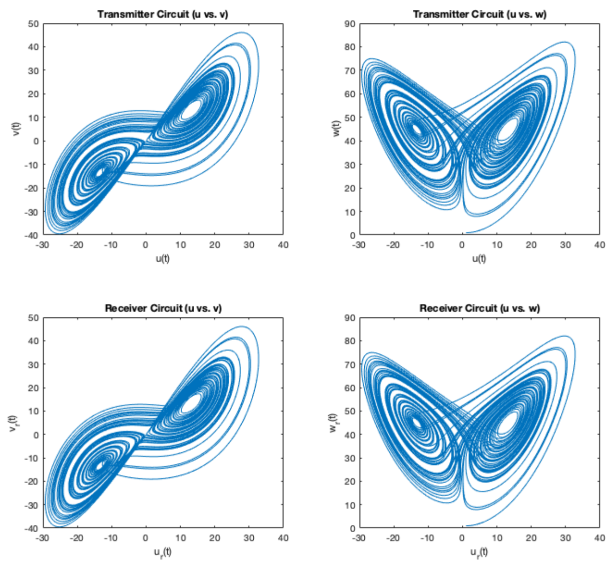

Fig. 3: Transmitter initial values are $u_0 = 1$, $v_0 = 1$, $w_0 = 1.05$

Looking the same isn't good enough to really verify these signals are indeed the same. Since we have two systems, we will have two sets of trajectories. Let the trajectory for the transmitter be the vector $\vec{T} =< u(t), v(t), w(t) >$. Let the trajectory for the receiver be $\vec{R} =< u_r(t), v_r(t), w_r(t) >$. If they are in sync, then we should expect the trajectories to be the same, that is: $|\vec{T} - \vec{R}| = \vec{0}$ as we've described previously. When we plot $w_r(t)$ against$w(t)$, and $v_r(t)$ against $v(t)$ then we should expect to see a straight line.

We can see that the individual trajectories on the transmitter and receiver signals line up perfectly, which indeed shows the two systems are in sync (Fig 4).

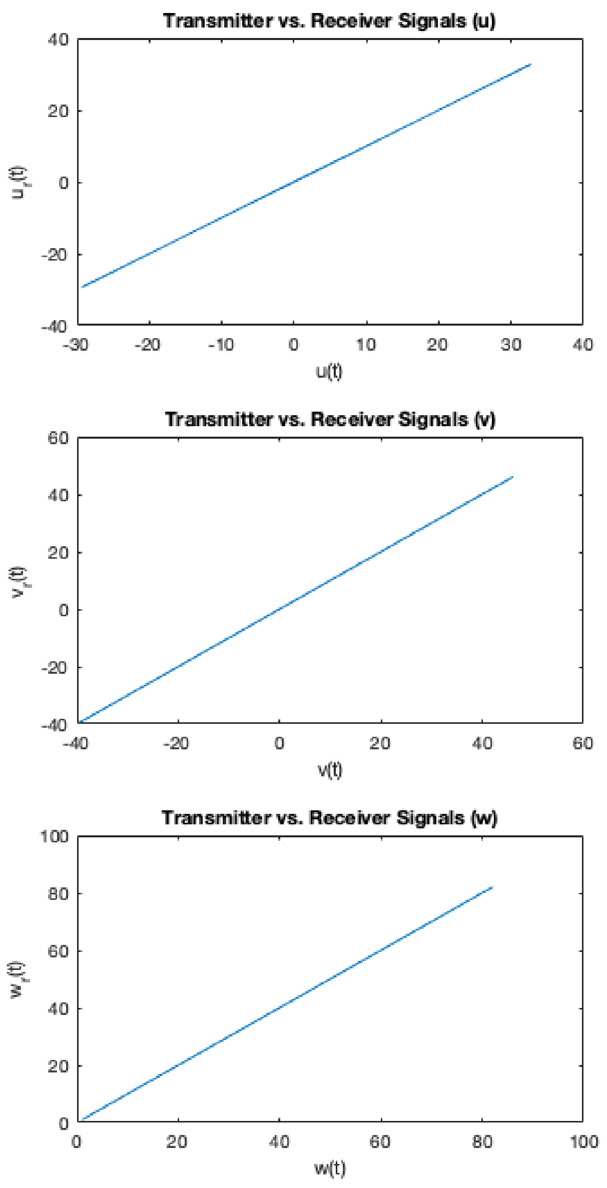

Fig. 4: Respective transmitter and receiver trajectories are plotted against each other, which resemble a 45 degree line.

An interesting point is that even perturbing the receiver slightly starting at $v_0 = -1$, we see the plots remain fairly the same. We show this in figure 5, but only include the plot for $v(t)$ simply to save space here. In any case, this is quite remarkable! This shows the synchronization of the chaos in the two Lorenz systems is fairly robust against small errors in the driving signal.

When the circuit in Fig. 6 is used in real-life applications, wiring to the receiver may be jerked, data may be jumbled, and signal precision may be lost. This is why it is so important for the synchronization to be as stable as it is.

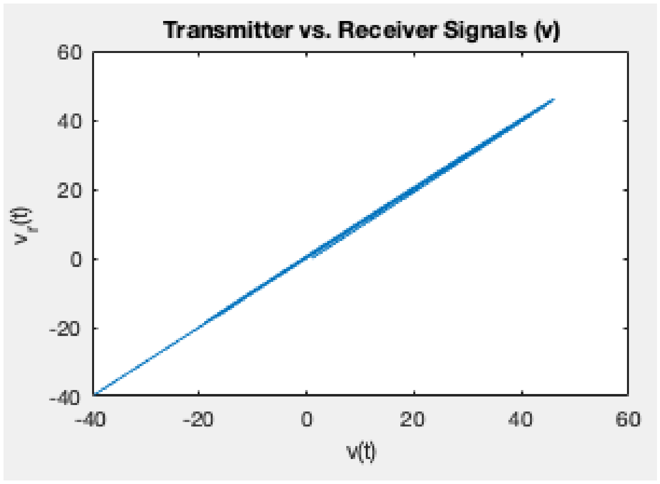

Fig. 5: The trajectories still line up well enough despite a slightly perturbation

The small experiments we ran here show the feasibility of actually synchronizing these systems. On the other hand, this means an attacker could reconstruct the waveform relatively well enough with close enough parameters and/or initial conditions.

V. Chaotic Signal Masking

Up to this point, we've described a clever mechanism for exchanging waveforms between a transmitter and receiver. But we can use this technique to send a secret message such that the receiver can recover the secret. We now introduce the concept of signal masking, which is where the Lorenz equation synchronization technique really shines. Suppose Bob wants to send a waveform to Alice, but neither of them wants Eve to eavesdrop on their waveform, since it is sent over a public channel. This is where signal masking comes in - Bob uses the output of his sender circuit, $u(t)$ to add noise to the waveform he intends to send, $m(t)$. We will call his new waveform $s(t)$. Alice will receive $s(t)$ when it arrives from the public channel. When it's fed into her receiver circuit, she gets $u_r(t)$, then subtracts that from $s(t)$ to get back the original signal: $\hat{m}(t) = s(t) - u_r(t)$.

In this case, the transmitter circuit is the same as system (6) - nothing changes. We wrote MATLAB code to simulate the transmitter circuit for a finite time, which added $m(t)$ (Bob's information-bearing signal) to the transmitter's output, $u(t)$ (the mask). Since $s(t) = u(t) + m(t)$, in this particular case, the receiver circuit equations are:

$$\begin{aligned} \dot{u_r} &= \sigma(v_r - u_r) \\ \dot{v_r} &= rs(t) - v_r - 20s(t)w_r \\ \dot{w_r} &= 5s(t)v_r - bw_r \end{aligned} \tag{9}$$

Notice the main difference between this and the previous scheme the substitution of $s(t)$ for $u(t)$ in the receiver circuit. $s(t)$ is the driving waveform to the receiver now. The reasoning for this is that the data-carrying waveform $m(t)$ has significantly less amplitude and frequency than

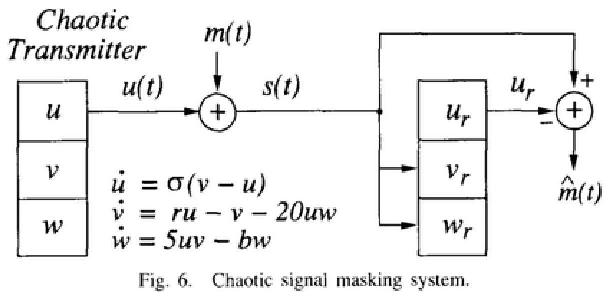

Fig. 6: Cuomo's original circuit diagram for signal masking.

the chaotically-generated waveform $s(t)$. Thus, the receiver system gives us a result for $u_r(t)$ that is very similar to the original $u(t)$, despite some disturbance.

We had our MATLAB code then compute the recovered signal, $\hat{m}(t)$, as best it could. In this case, $m(t) = sin(t)$ is the information-bearing signal. We plot this against the recovered signal $\hat{m}(t)$ and show how the qualitative shape is similar. We would require down-sampling and some smoothing to improve the accuracy of the recovered signal.

Why does this work? As we've discussed, if systems (6) and (9) are synchronized, then we know $u(t) \approx u_r(t)$ from the previous section. This implies $\hat{m}(t) \approx m(t)$. Consequently, as time tends to infinity, the error will tend to zero.

Figure 7 shows how $\hat{m}(t)$ is similar to $m(t)$. In this scheme, we've used the same parameters: $\sigma = 16$, $r = 45.6$, $b = 4$, and chosen $m(t) = sin(t)$ along the domain $t \in [0, 4\pi]$. Seeing how the signals line up is quite satisfying in our generated plot.

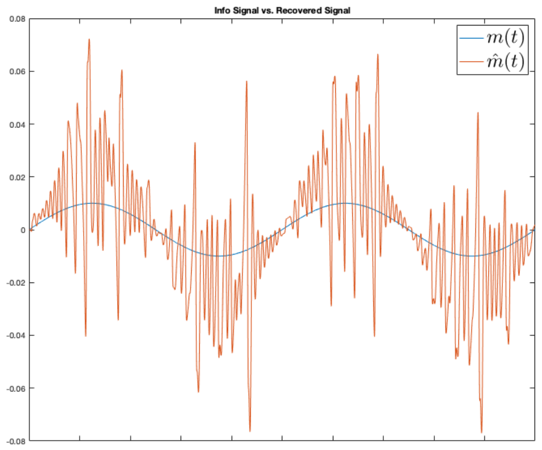

Fig. 7: Original sine signal and recovered signal

VI. Binary Signals

Waveforms have a very large range, so ensuring they are exact may be quite challenging if downsampling and smoothing aren't done properly. Rather than sending audio waveforms, we can actually encode binary in a waveform instead, where 1 would be high and 0 would be low. In this case, a human could make out the original signal much easier, since there are only two states. In this case, the sender and receiver systems become the following:

$$\begin{aligned} \dot{u} &= \sigma(v-u) \\ \dot{v} &= ru - v - 20uw \\ \dot{w} &= 5uv - B(m(t))w \end{aligned} \tag{10}$$

$$\begin{aligned} \dot{u_r} &= \sigma(v_r-u_r) \\ \dot{v_r} &= ru - v_r - 20uw_r \\ \dot{w_r} &= 5uv_r - bw_r \end{aligned} \tag{11}$$

In system 10, we encode a binary wave, and insert it into $\dot{w}(t)$, where

$$B(m(t)) = \begin{cases} 4.4 & \text{sending 1 bit} \quad m(t)=1 \\ 4 & \text{sending 0 bit} \quad m(t)=0 \end{cases}$$

The resulting $u(t)$ is used as the driving signal in system (11), as usual. Note in system (11) that $\dot{w_r}(t)$ includes the parameter $b = 4$, **not** our binary wave $B(t)$.

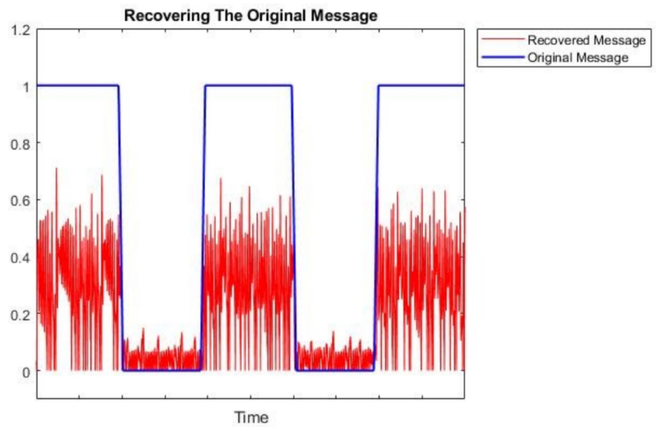

Fig. 8: Original Square Wave signal and recovered signal

We've again written some MATLAB code which shows how the signals match up. The blue wave is the original binary signal, whereas the red signal is the recovered signal (Fig. 8). Smoothing this red wave out would be significantly easier than in the previous method which involved regular signal masking. Furthermore, without smoothing, a human is able to interpret the recovered wave much easier.

VII. Further Applications: Image Encryption

We can also apply this principle of using one circuit to drive another to existing chaotic encryption systems. Here, we have taken an existing image encryption system taken from the MATLAB forums (authored by GodGOD) and modified it to use a driven system. The code encrypts an image by permuting the pixels based on a trajectory for a specific Lorenz system with specific initial conditions.

In figure 9b, we encrypted Lena using the aforementioned algorithm and the Lorenz system with settings: $\sigma = 10, r = 28, b = \frac{8}{3}$; $x_0 = 1.1840, y_0 = 1.3627, z_0 = 1.2519$.

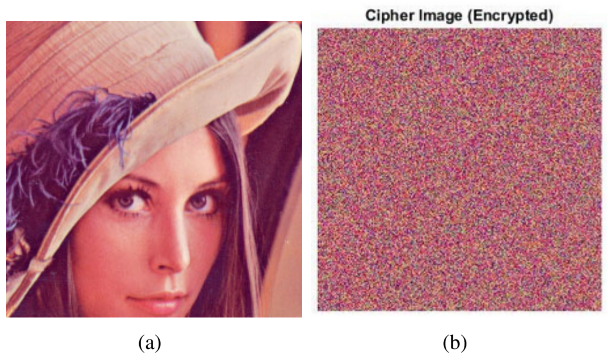

Fig. 9: (a) Lena in all her glory (b) Encrypted image

Now, we use the solution $x(t)$ as the driver to the receiver circuit. Using the exact same parameters and initial values, we can produce precisely the same image when we run the algorithm in reverse. This is shown in figure 10b.

What happens if the initial values are close to what they should be? If we use the original system that is not being driven, we just get noise back. However, when we use a driven system, we can still recover a large amount of information from the image. In terms of reconstructing a digital image, this translates into something fuzzy. This is shown in figure 11b. The receiver circuit used the following

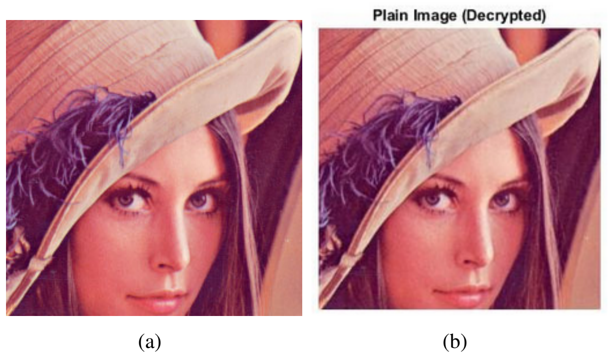

Fig. 10: (a) Lena in all her glory (b) Decrypted image

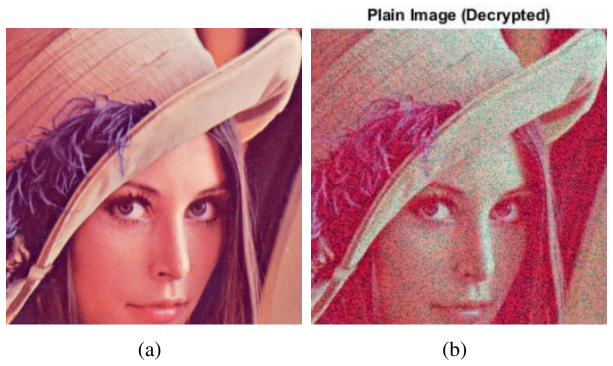

Fig. 11: (a) Lena in all her glory (b) Fuzzy, partially decrypted image

settings: $\sigma = 10, r = 28, b = \frac{8}{3}$; $x_0 = 1.6840, y_0 = 1.3627, z_0 = 1.2519$. The **only** difference is that x_0 has been perturbed by $\frac{1}{2}$ at the receiver, and the driver does its best to synchronize the systems back together, but they are not quite in sync.

To account for this close correlation to the plain image, we could experiment how this image behaves with signal masking as in the previous section. The real challenge would be smoothing the recovered signal to ensure the image indeed looks proper.

VIII. Remarks

Note that this system is designed in such a way that it can be implemented in physical hardware: The hardware just needs to be able to run the Lorenz system and be constructed such that one system drives the other. This means that this form of security can exist on hardware itself, instead of just in software.

We demonstrated driving an existing simple Lorenz encryption system, however, more complicated encryption systems based on the chaos on the Lorenz system exist [1]. This method is a much stronger form of encryption than the one we presented, and as a result may not benefit from being altered to a driven transmitter-receiver system.

Overall, this paper is a very interesting glimpse into signal processing, masking, and chaotic synchronization. The many applications make the simplicity and elegance of the design and analysis all that much more impressive!

All of our code is here: https://github.com/ericsund/467-chaos-theory.

IX. Bibliography

References

[1] Kayhan CELIK and Erol KURT. “A New Image Encryption Algorithm Based on Lorenz System”. In: *International Journal of Information and Education Technology* 1.2 (2011).

[2] K.M. Cuomo, A.V. Oppenheim, and S.H. Strogatz. “Synchronization of Lorenz-based chaotic circuits with applications to communications”. In: *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing* 40.10 (1993), pp. 626–633. DOI: 10.1109/82.246163.