Public channel cryptography by synchronization of neural networks and chaotic maps

Rachel Mislovaty,[1] Einat Klein,[1] Ido Kanter,[1] and Wolfgang Kinzel[2]
[1]*Department of Physics, Bar Ilan University, Ramat-Gan 52900, Israel*
[2]*Institute of Theoretical Physics, University of Würzburg, D-97074 Würzburg, Germany*

Two different kinds of synchronization have been applied to cryptography: Synchronization of chaotic maps by one common external signal and synchronization of neural networks by mutual learning. By combining these two mechanisms, where the external signal to the chaotic maps is synchronized by the nets, we construct a hybrid network which allows a secure generation of secret encryption keys over a public channel. The security with respect to attacks, recently proposed by Shamir et al, is increased by chaotic synchronization.

PACS numbers: PACS numbers: 05.45.Gg, 05.45.Vx, 87.18.Sn

Two identical dynamical systems, starting from different initial conditions, can be synchronized by a common external signal which is coupled to the two systems [1]. It has been shown that even chaotic systems can be synchronized although the correlation between external signal and the common dynamics still remains chaotic [2]. This phenomenon has been applied to *private-key* cryptography: If two partners A and B want to exchange a secret message, A adds her message to a synchronized signal while B subtracts it. Of course, A and B need a common secret (private key), namely, the algorithm and the parameters of their identical chaotic system.

Synchronization has recently been observed in artificial neural networks as well. Two networks which are trained on their mutual output can synchronize to a time-dependent state of identical synaptic weights [3]. This phenomenon has been applied to cryptography as well [4]. In this case, the two partners A and B do not have to share a common secret but use their identical weights as a secret key needed for encryption. The secret key is generated over a *public channel*. An attacker E who knows all the details of the algorithm and records any communication transmitted through this channel finds it difficult to synchronize with the parties, and hence to calculate the common secret key. Synchronization by mutual learning (A and B) is much faster than learning by listening (E).

Neural cryptography is much simpler than the commonly used algorithms which are mainly based on number theory [5] or on quantum mechanics [6]. In fact, it can be expressed as synchronization of an ensemble of random walks with reflecting boundaries [7]. But the question remains: Is it secure? Does an algorithm exist which can decipher the secret key from the transmitted information? For the set of parameters used in Ref. [4] it has been shown that such algorithms do exist [8]. In an ensemble of attackers there is a nonzero chance that some of them will synchronize to the two partners. However, it has recently been shown that the probability of a successful attack can be made exponentially small [9]; it decreases like $\exp(-yL)$ where the parameter L (stands for the depths of the weights of the networks) is defined

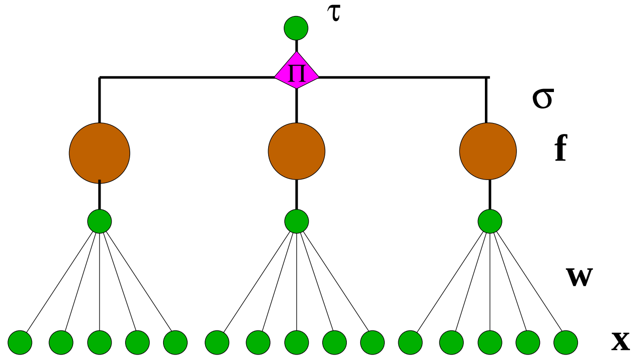

FIG. 1: Parity machine for $K = 3$ with chaotic map f.

below. Hence for large values of L the computational time is so long that an attack is infeasible, meaning that neural cryptography remains secure. Of course, similar to classic key-exchange protocols, one cannot prove that there does not exist any other algorithm which cracks the system.

In this Letter we combine neural cryptography with chaotic synchronization. Both partners A and B use their neural networks as input for the logistic maps which generate the output bits to be learned. By mutually learning these bits, the two neural networks approach each other and produce an identical signal to the chaotic maps which – in turn – synchronize as well, therefore accelerating the synchronization of the neural nets.

We show that the security of key generation increases as the system approaches the critical point of chaotic synchronization, and it is possible that the exponent y diverges as the coupling constant between the neural nets and the chaotic maps is tuned to be critical.

We start with the parity machine (PM) with K hidden units which are arranged in a tree architecture as shown in figure 1 for $K = 3$. Each hidden unit has N discrete weights $w_{k,j}$ which can take the values $\{-L, -L+1, ...L-1, L\}$. At every training step the network receives an input vector consisting of KN components $x_{k,j} \in \{+1, -1\}$. Each hidden unit generates a local field

$$h_k = \sum_{j=1}^{N} w_{kj} x_{kj} \tag{1}$$

Previously, the output bit of each hidden unit was the sign of the local field[4]. Now we combine the PM with chaotic synchronization by feeding the local fields into logistic maps:

$$s_k(t+1) = \lambda(1-\beta)s_k(t)(1-s_k(t)) + \frac{\beta}{2}\tilde{h}_k(t) \tag{2}$$

Here $\tilde{h}$ denotes a transformed local field which is shifted and normalized to fit into the interval $[0,2]$[11]. For $\beta = 0$ one has the usual quadratic iteration which produces K chaotic series $s_k(t)$ when the parameter λ is chosen correspondingly; in this Letter we use $\lambda = 3.95$. For $0 < \beta < 1$ the logistic maps are coupled to the fields of the hidden units. It has been shown that such a coupling leads to chaotic synchronization[2]: If two identical maps with different initial conditions are coupled to a *common* external signal they synchronize when the coupling strength is large enough, $\beta > \beta_c$.

Now we consider key generation between two partners A and B. Each partner uses a PM with logistic maps. Hence each partner has a time series of KN weights $w_{kj}^{A/B}(t)$, K local fields $h_k^{A/B}(t)$ and K signals $s_k^{A/B}(t)$. In addition, a common external sequence of random inputs $x_{kj}(t)$ is presented to both of the partners. This sequence of inputs is public, as well as the complete architecture and the parameters β and λ. Each partner generates random initial weights $w_{kj}^{A/B}(t=0)$ which are not public and not known to each other.

In the original version of neural cryptography [4] the synchronization of the weights, $w_{kj}^A(t) = w_{kj}^B(t)$ for $t > t_{sync}$ was achieved by training, for instance, in the simplest symmetric version the training step reads

$$w_{kj}^A(t+1) = w_{kj}^A(t) - x_{kj}(t)\tau_k(t) \tag{3}$$

for partner A and the same for partner B and $\tau_k(t)$ is defined in eq. 4. When a weight moves outside of the allowed interval it is reset to the corresponding boundary value $\pm L$. Note that the equation above may be considered as a random walk with reflecting boundaries.

The security of synchronization is achieved by the parity construction. The training step is performed only if the output bits τ^A, τ^B of the two PMs are identical and, in addition, if the output bit σ_k^A of the hidden unit is identical to τ^A. In the parity network one defines

$$\tau^{A/B}(t) = \prod_{k=1}^{K} \sigma_k^{A/B}(t) \tag{4}$$

The output bits (τ^A, τ^B) which are transmitted at each training step generate control signals which produce a mixture of attractive, repulsive and quiet movements of the corresponding hidden units of A and B. Only the parity construction gives a low probability of repulsive steps compared to an attacker PM close to synchronization [10].

In the hybrid network introduced here, we keep the parity mechanism but we define the hidden output bits $\sigma_k^{A,B}$ by the signals $s_k^{A/B}$ of the logistic maps coupled to neural networks:

$$\sigma_k^A(t) = \mathrm{sign}(s_k^A(t) - s_0(\beta)) \tag{5}$$

The public parameter $s_0(\beta)$ is chosen such that σ takes the values ± 1 with equal probability.

Now the complete algorithm for the two partners A and B is defined. The parameter β controls the coupling strength between neural network and chaotic map. For $\beta = 1$ we obtain the PM studied previously [4, 10]. The two networks synchronize to common time dependent weights $w^A(t) = w^B(t)$. The average synchronization time t_{sync} scales with the size of the input as $\ln N$, and is therefore relatively short even for large systems. The synchronization time also increases as L is increased, and for $L < O(\sqrt{N})$ one finds that t_{sync} increases with L^2, as expected from random walk theory [9].

For $\beta = 0$ the two chaotic signals $(s_k^A(t), s_k^B(t))$ are not coupled and just generate random outputs σ and τ. As a consequence, the two networks do not synchronize.

By construction, the synchronized state

$$s_k^A(t) = s_k^B(t); \quad w_{kj}^A(t) = w_{kj}^B(t) \tag{6}$$

is a fixed point of the dynamics. The question remains: is it an attractor? In our model synchronization occurs by two mechanisms simultaneously. The weights of the two neural nets move towards a common sequence and the signals of the corresponding chaotic maps move towards a common chaotic sequence triggered by the local fields of the networks. Hence it is not at all obvious that synchronization is possible. Our numerical simulations as well as our analytic[13] calculations show that the two networks synchronize when the parameter β is larger than its critical value. The critical value, β_c, is defined such that the average synchronization time, t_{av}, diverges. Figure 2 presents the average synchronization time as a function of β for $K = 1,\ 2$ and $L = 10$. Result indicate that for $K = 1,\ 2$ $\beta_c \sim 0.15,\ 0.35$. Note that for $K = 1$ β_c is very close to the reported result for the synchronization of two logistic maps using common white signal[2], instead of gaussian signal, h, is eq. 2.

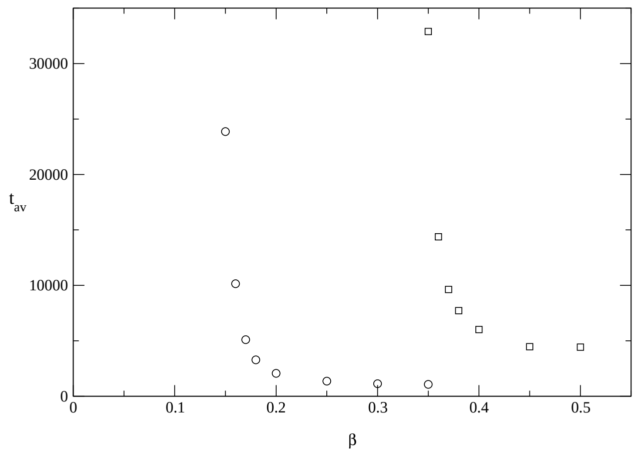

FIG. 2: The average synchronization time as a function of β, for L=10, N=1000, K=1 ($\circ$) and K=2 ($\Box$). Results were averaged over 1000 samples. Synchronization here is defined

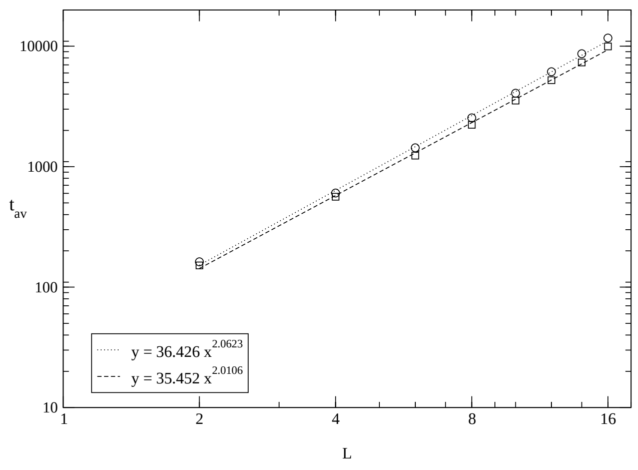

FIG. 3: Average synchronization time as a function of L for K=2, N=10000, β=0.45 ($\circ$), β=1 ($\Box$) and the regression power-law fit β=0.45 (dotted line) and β=1 (dashed line). Results were averaged over 1000 samples.

Figure 2 shows that the synchronization time as a function of L does not depend much on the parameter β. The average synchronization time, t_{av}, is almost constant for $0.45 < \beta < 1$, for all values of L studied.

Now we turn to the problem: Is the observed synchronization secure? Consider an attacker (eavesdropper E) who records the exchange of the bits $(\tau^A(t), \tau^B(t))$ and who knows the sequence $x_k(t)$ as well as the parameters of the hybrid networks. Can E calculate the common weights before the synchronization of A and B?

The most successful attack reported by the group of Shamir [8] is the flipping attack. We generalize this attack to our hybrid network as follows. The attacker E uses a network identical to the ones of A and B and trains its weights only if the output bits of A and B agree.

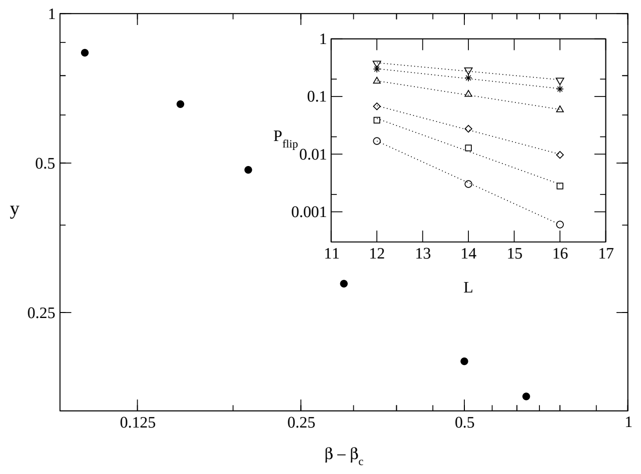

FIG. 4: The exponent y as a function of β-β_c, where β_c=0.35 (see Fig. 2). In the inset P_{flip} as a function of L is plotted for different values of β to obtain y. The results are for K=2, N=10000 and averaged over 10000 attackers. The different lines represent (from top to bottom) β=1, 0.85, 0.65, 0.55, 0.5, 0.45. The slopes of the fitted lines are respectively: -0.169, -0.199, -0.286, -0.485, -0.657, -0.834.

When τ^E agrees with τ^A and τ^B the attacker E learns its weights as defined before, eq. (3)[10]. However, when $\tau^E \neq \tau^A = \tau^B$ the hidden unit with the "weakest" local field $\tilde{h}_k^E$ is selected and the sign of its output bit σ_k^E is changed. With this redefined hidden unit learning proceeds as usual. The weakest field is the one which has the smallest distance to the decision boundary given by equations (2) and (5).

Figure 4 shows the main result of this Letter. The inset of Figure 3 indicates that the probability of a successful attack decreases exponentially fast with the level number L,

$$P_{flip} = A \exp(-yL) \qquad (7)$$

But contrary to the synchronization time, this probability has a strong dependence on the coupling strength β. Figure 4 shows that the attacker's success rate P_{flip}/y decreases/increases as β approaches the critical value, $\beta_c \sim 0.35$, from above. It is clear from 4 that y increases as β approaches β_c from above, and it is possible that y diverges close to criticality. However from the current data we cannot rule out other scenarios including the one that y is finite at criticality and a further investigation of this question is required.

Note that the values of β in our simulations are still far away from the critical point. For a fixed size of the system N and close to β_c the synchronization time increases beyond the scaling reported in Fig. 2. Hence a finite network is not useful for key generation close to β_c. Anyway, the main result is that the security of the network strongly increases when the hidden units are screened by chaotic synchronization. For example: The synchronization time of a single attacker scales like $L^2 N \ln N$. We need about $\exp(yL)$ attackers on the average to be successful. If we can use one year of a teraflop computer for each message, we have about 10^{20} calculations available. Hence, for $N = 10^5$, we need a level number of about $L \sim 135$ without chaotic synchronization, $\beta = 1$. For $\beta = 0.45$, however, we need a value of $L \sim 25$, only. It indicates that the two partners A and B need less than 5% of training steps to synchronize in comparison to the same system without chaotic synchronization ($\beta = 1$).

Finally, we note that the effect of the chaotic map on a hidden unit is essentially the generation of noise. We have replaced the chaotic map in equation (2) by randomly flipping the output bit $\sigma = \mathrm{sign}\, h$ with some probability p measured in actual simulation. This probability, p, is suppressed to zero as the two neural networks approach each other. For this approximation we can solve the dynamics of synchronization analytically, by using the methods of Ref. [10, 12]. We find good agreement between the noise approximation and the actual simulations using the chaotic maps[13].

[1] For a review see: A. Pikovsky, M. Rosenblum and J. Kurths, *Synchronization*, (Cambridge University Press 2001)

[2] L. M. Pecora and T. L. Carroll, Phys. Rev. Lett. **64** 821-824, (1990); K.M. Cuomo and A.V. Oppenheim, Phys. Rev. Lett. **71** 65 (1993); C.M. Kim, S. Rim, W.H. Kye, Phys. Rev. Lett. **88** 014103 (2002);

[3] R. Metzler and W. Kinzel and I. Kanter, Phys. Rev. E **62**, 2555 (2000)

[4] I. Kanter, W. Kinzel and E. Kanter, Europhys. Lett. **57**, 141-147 (2002)

[5] D. R. Stinson, *Cryptography: Theory and Practice* (CRC Press 2002)

[6] C.P. Williams and S.H. Clearwater, *Explorations in Quantum Computing* (Springer Verlag, 1998)

[7] W. Kinzel and I. Kanter, Advances in Solid State Physics **42**, 383, (Springer Verlag, 2002)

[8] A. Klimov, A. Mityagin and A. Shamir, ASIACRYPT 2002

[9] R. Mislovaty, Y. Perchenok, I. Kanter and W. Kinzel, Phys. Rev. E **66**, 066102 (2002)

[10] M. Rosen-Zvi, E. Klein, I. Kanter and W. Kinzel, Phys. Rev. E **66**, 066135 (2002)

[11] In our simulations we define $\tilde{h}_k(t) = (1 + h_k(t)/(2.5 h_{av}))$ where h_{av} is defined as the average field over the last given T_0 step. In our simulations T_0=20. In the event $|\tilde{h}_k(t)| > 1$ then $\tilde{h} = 1 + sign(h)$.

[12] M. Rosen-Zvi, I. Kanter and W. Kinzel, cond-mat/0202350 (2002)

[13] R. Mislovaty, E. Klein, I. Kanter and W. Kinzel (unpublished).