Genetic attack on neural cryptography

Andreas Ruttor and Wolfgang Kinzel
Institut für Theoretische Physik, Universität Würzburg, Am Hubland, 97074 Würzburg, Germany

Rivka Naeh and Ido Kanter
Minerva Center and Department of Physics, Bar Ilan University, Ramat Gan 52900, Israel
(Received 1 December 2005; published 17 March 2006)

Different scaling properties for the complexity of bidirectional synchronization and unidirectional learning are essential for the security of neural cryptography. Incrementing the synaptic depth of the networks increases the synchronization time only polynomially, but the success of the geometric attack is reduced exponentially and it clearly fails in the limit of infinite synaptic depth. This method is improved by adding a genetic algorithm, which selects the fittest neural networks. The probability of a successful genetic attack is calculated for different model parameters using numerical simulations. The results show that scaling laws observed in the case of other attacks hold for the improved algorithm, too. The number of networks needed for an effective attack grows exponentially with increasing synaptic depth. In addition, finite-size effects caused by Hebbian and anti-Hebbian learning are analyzed. These learning rules converge to the random walk rule if the synaptic depth is small compared to the square root of the system size.

DOI: 10.1103/PhysRevE.73.036121 PACS number(s): 84.35.+i, 87.18.Sn, 89.70.+c

I. INTRODUCTION

Neural cryptography [1,2] is based on the effect that two neural networks are able to synchronize by mutual learning [3,4]. In each step of this online learning procedure they receive a common input pattern and calculate their output. Then, both neural networks use those outputs presented by their partner to adjust their own weights. So, they act as teacher and student simultaneously. Finally, this process leads to fully synchronized weight vectors.

Synchronization of neural networks is, in fact, a complex dynamical process. The weights of the networks perform random walks, which are driven by a competition of attractive and repulsive stochastic forces [5]. Two neural networks can increase the attractive effect of their moves by cooperating with each other. But, a third network which is only trained by the other two clearly has a disadvantage, because it cannot skip some repulsive steps. Therefore, bidirectional synchronization is much faster than unidirectional learning [2].

This effect can be applied to solve a cryptographic problem: Two partners A and B want to exchange a secret message. A encrypts the message to protect the content against an opponent E, who is listening to the communication. But, B needs A's key in order to decrypt the message. Therefore, the partners have to use a cryptographic key-exchange protocol [6] in order to generate a common secret key. This can be achieved by synchronizing two neural networks, one for A and one for B, respectively. The attacker E trains a third neural network using inputs and outputs transmitted by the partners as examples. But, on average, learning is slower than synchronization. Thus, there is only a small probability P_E that E is successful before A and B synchronize [2].

While other cryptographic algorithms use complicated calculations based on number theory [6], the neural key-exchange protocol only needs basic mathematical operations, namely adding and subtracting integer numbers. These can be realized efficiently in integrated circuits. Computer scientists are already working on an hardware implementation of neural cryptography [7–10].

Since the first proposal [1] of the neural key-exchange protocol, improved strategies for the attackers [11,12] and the partners [5,13,14] have been suggested and analyzed [2,15–17]. For the *geometric attack* it has been found that the synaptic depth L determines the security of the system: the success probability P_E decreases exponentially with L, while the synchronization time t_{sync} increases only proportionally to L^2 [17,18]. Therefore, any desired level of security against this attack can be reached by increasing L.

An improved version of this method is the *majority attack* [12]. Here, a group of M neural networks estimates the output of B's hidden units. But, instead of updating the weights individually, E's tree parity machines cooperate and adjust the weight vectors in the same way according to the majority vote. While using this method increases P_E, the scaling laws hold except for one special learning rule and random inputs [12,14]. Therefore, neural cryptography is secure against this attack in the limit $L \to \infty$, too.

In this paper we analyze a different method for the opponent E. The *genetic attack* [11] is not based on optimal learning like the majority attack [12], but employs a genetic algorithm in order to select the most successful of E's neural networks. First, we repeat the definition of the neural key-exchange protocol in Sec. II. We also explain why A and B have a clear advantage over E. The algorithm of the genetic attack is presented in Sec. III. Here, we show that the scaling behavior observed for the geometric attack and the majority attack also holds for the genetic attack. In Sec. IV we analyze the influence of the learning rules on synchronization and learning. Finally, the known attacks on the neural key-exchange protocol are compared regarding their efficiency. The results presented in Sec. V show that the genetic attack is less efficient than the majority attack except for some special cases.

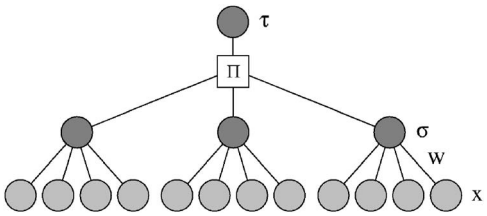

FIG. 1. Tree parity machine with $K=3$ and $N=4$.

II. NEURAL CRYPTOGRAPHY

In this section we repeat the definition of the neural key-exchange protocol [1]. Each partner, A and B, uses a tree parity machine. The structure of this neural network is shown in Fig. 1. A tree parity machine consists of K hidden units, which work like perceptrons. The possible input values are binary,

$$x_{ij} \in \{-1, +1\}, \tag{1}$$

and the weights are discrete numbers between $-L$ and $+L$,

$$w_{ij} \in \{-L, -L+1, \ldots, L-1, L\}. \tag{2}$$

Here, the index $i=1,\ldots,K$ denotes the ith hidden unit of the tree parity machine and $j=1,\ldots,N$ the elements in each vector. The output of the first layer is defined as the sign of the scalar product of inputs and weights,

$$\sigma_i = \mathrm{sgn}(\mathbf{w}_i \cdot \mathbf{x}_i). \tag{3}$$

And, the total output of the tree parity machine is given by the product (parity) of the hidden units,

$$\tau = \prod_{i=1}^{K} \sigma_i. \tag{4}$$

At the beginning of the synchronization process A and B initialize the weights of their neural networks randomly. This initial state is kept secret. In each time step t, K random input vectors $\mathbf{x}_i$ are generated publicly and the partners calculate the outputs τ^A and τ^B of their tree parity machines. After communicating the output bits to each other they update the weights according to one of the following learning rules:

(i) Hebbian learning

$$\mathbf{w}_i^+ = \mathbf{w}_i + \sigma_i \mathbf{x}_i \Theta(\sigma_i \tau) \Theta(\tau^A \tau^B), \tag{5}$$

(ii) Anti-Hebbian learning

$$\mathbf{w}_i^+ = \mathbf{w}_i - \sigma_i \mathbf{x}_i \Theta(\sigma_i \tau) \Theta(\tau^A \tau^B), \tag{6}$$

(iii) Random walk

$$\mathbf{w}_i^+ = \mathbf{w}_i + \mathbf{x}_i \Theta(\sigma_i \tau) \Theta(\tau^A \tau^B). \tag{7}$$

If any component of the weight vectors moves out of the range $-L,\ldots,+L$, it is replaced by the nearest boundary value, either $-L$ or $+L$.

After some time t_{sync} the partners have synchronized their tree parity machines, $\mathbf{w}_i^A(t_{\mathrm{sync}}) = \mathbf{w}_i^B(t_{\mathrm{sync}})$, and the process is stopped. Afterwards, A and B can use the weight vectors as a common secret key in order to encrypt and decrypt secret messages.

We describe the process of synchronization by standard order parameters, which are also used for the analysis of online learning [19]. These order parameters are

$$Q_i^m = \frac{1}{N} \mathbf{w}_i^m \cdot \mathbf{w}_i^m, \tag{8}$$

$$R_i^{m,n} = \frac{1}{N} \mathbf{w}_i^m \cdot \mathbf{w}_i^n, \tag{9}$$

where the indices $m,n \in \{A,B,E\}$ denote A's, B's, or E's tree parity machine, respectively. The level of synchronization between two corresponding hidden units is defined by the (normalized) overlap,

$$\rho_i^{m,n} = \frac{\mathbf{w}_i^m \cdot \mathbf{w}_i^n}{\sqrt{\mathbf{w}_i^m \cdot \mathbf{w}_i^m}\sqrt{\mathbf{w}_i^n \cdot \mathbf{w}_i^n}} = \frac{R_i^{m,n}}{\sqrt{Q_i^m Q_i^n}}. \tag{10}$$

Uncorrelated weight vectors have $\rho=0$, while the maximum value $\rho=1$ is reached for full synchronization.

The overlap between two corresponding hidden units increases if the weights of both neural networks are updated in the same way. Coordinated moves, which occur for identical σ_i, have an attractive effect.

Changing the weights in only one hidden unit decreases the overlap on average. These repulsive steps can only occur if the two output values σ_i are different. The probability for this event is given by the well-known generalization error of the perceptron [19],

$$\epsilon_i = \frac{1}{\pi} \arccos \rho_i, \tag{11}$$

which itself is a function of the overlap ρ_i between the hidden units. For an attacker who simply trains a third tree parity machine using the examples generated by A and B, repulsive steps occur with probability $P_r^E = \epsilon_i$, because E cannot influence the process of synchronization.

In contrast, A and B communicate with each other and are able to interact. If they disagree on the total output, there is at least one hidden unit with $\sigma_i^A \neq \sigma_i^B$. As an update would have a repulsive effect, the partners just do not change the weights. In doing so, A and B reduce the probability of repulsive steps in their hidden units. For $K=3$ and identical generalization error, $\epsilon_i = \epsilon$, we find [5]

$$P_r^B = \frac{2(1-\epsilon)\epsilon^2}{(1-\epsilon)^3 + 3(1-\epsilon)\epsilon^2} \leqslant \epsilon = P_r^E. \tag{12}$$

Therefore, the partners have a clear advantage over an attacker using only simple learning.

But, E can use a more advanced method called *geometric attack*. As before, she trains a third tree parity machine, which has the same structure as A's and B's. In each step τ^E is calculated and compared to τ^B. As long as these output values are identical, E can apply the learning rule in the same manner as B. But, if $\tau^E \neq \tau^B$, the attacker has to correct this deviation before updating the weights.

For this purpose E uses the local field

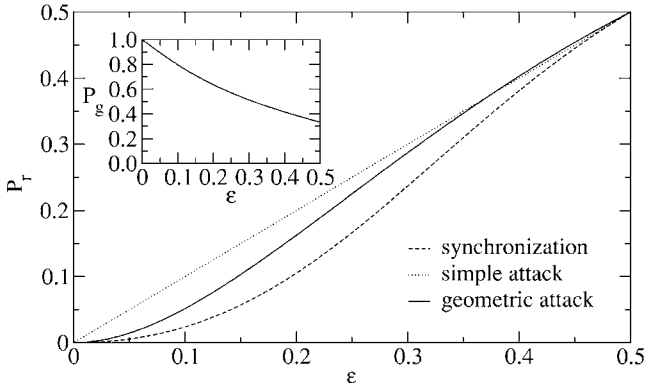

FIG. 2. Probability P_r of repulsive steps as a function of the generalization error ϵ. The inset shows the probability P_g for a successful geometric correction.

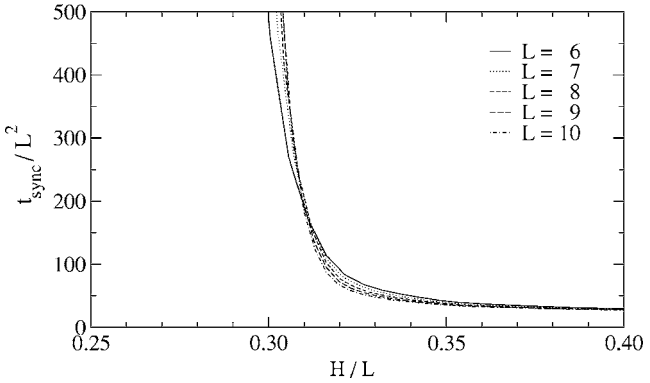

FIG. 3. Synchronization time t_{sync} as a function of H for $K=3$, $N=1000$, random walk learning rule, and different values of L, averaged over 10 000 simulations.

$$h_i = \frac{1}{\sqrt{N}} \mathbf{w}_i \cdot \mathbf{x}_i \tag{13}$$

of her hidden units as additional information. Then, the probability of $\sigma_i^B \neq \sigma_i^E$ is given by the prediction error of the perceptron [20],

$$\epsilon_i(\rho_i, h_i) = \frac{1}{2}\left[1 - \text{erf}\left(\frac{\rho_i}{\sqrt{2(1-\rho_i^2)}} \frac{|h_i|}{\sqrt{Q_i}}\right)\right]. \tag{14}$$

If the local field h_i is zero, the neural network has no information about the input vector $\mathbf{x}_i$, because it is perpendicular to the weight vector $\mathbf{w}_i$. In this case the prediction error reaches its global maximum of $\epsilon_i(\rho_i, 0) = 1/2$.

The prediction error $\epsilon_i(\rho_i, h_i)$ is a strictly monotonic decreasing function of $|h_i|$. Therefore, the attacker searches the hidden unit with the lowest value of the absolute local field $|h_i^E|$ and flips the sign of σ_i^E. This results in $\tau^E = \tau^B$ and the learning rule can be applied. But, the geometric attack does not always find the correct hidden unit which caused the deviation of the total output bits. If $\sigma_i^E \neq \sigma_i^B$ in the ith hidden unit and $\sigma_j^E = \sigma_j^B$ in all other hidden units, E flips the sign of σ_i^E with probability

$$P_g = \int_0^\infty \prod_{j \neq i} \left(\int_{h_i}^\infty \sqrt{\frac{2\pi}{Q_j}} \frac{1 - \epsilon_j(\rho_j, h_j)}{\pi - \arccos \rho_j} e^{-h_j^2/(2Q_j)} \mathrm{d}h_j \right) \times \sqrt{\frac{2\pi}{Q_i}} \frac{\epsilon_i(\rho_i, h_i)}{\arccos \rho_i} e^{-h_i^2/(2Q_i)} \mathrm{d}h_i. \tag{15}$$

Thus, the geometric attacker avoids some repulsive steps, although they still occur more frequently than in the partners' tree parity machines.

In the case of identical generalization error $\epsilon_i = \epsilon$ and $K=3$, we find that the probability of repulsive steps,

$$P_r^E = 2(1 - P_g)(1 - \epsilon)^2 \epsilon + 2(1 - \epsilon)\epsilon^2 + \frac{2}{3}\epsilon^3, \tag{16}$$

is higher than P_r^B, but lower than $P_r^E = \epsilon$ for simple learning. This result is clearly visible in Fig. 2. That is why learning by listening is slower than mutual learning, even for advanced algorithms. This effect makes neural cryptography feasible and prevents successful attacks in the limit $L \to \infty$.

Recently, it has been discovered that the security of the neural key-exchange protocol can be improved by using queries instead of random inputs [14,21]. The partners ask questions to each other which depend on their own weight vectors $\mathbf{w}_i$ and an additional public parameter H. In odd (even) steps A (B) generates K input vectors $\mathbf{x}_i$ with $h_i^A \approx \pm H$ ($h_i^B \approx \pm H$). So, the absolute value of the local field h_i is given by H, while its sign σ_i is chosen randomly.

Queries change the relation between the overlap and the frequency P_r of repulsive steps. The probability of different outputs σ_i in corresponding hidden units is now given by Eq. (14) instead of Eq. (11), because the absolute local field in A's or B's hidden units is known. Consequently, the partners can optimize complexity and security of the neural key-exchange protocol by adjusting H and L suitably [14].

As shown in Fig. 3, a minimum value of H is needed in order to achieve synchronization in a reasonable number of steps. If $H > \alpha_c L$, t_{sync} increases proportional to $L^2 \ln N$, but for $H < \alpha_c L$ it diverges [14,18]. In the case of the random walk learning rule we estimate $\alpha_c \approx 0.31$ by using the extrapolation method described in [14].

III. GENETIC ATTACK

For the genetic attack [11] the opponent starts with only one tree parity machine, but she can use up to M neural networks. As before E calculates the output of her networks in each step. Afterwards, the following genetic algorithm is applied:

(i) If $\tau^A = \tau^B$ and E has at most $M/2^{K-1}$ tree parity machines, she determines all 2^{K-1} internal representations $(\sigma_1^E, \ldots, \sigma_K^E)$ which reproduce the output τ^A. Then, these are used to update the weights in E's neural networks according to the learning rule, so that 2^{K-1} variants of each tree parity machine are generated.

(ii) But, if E already has more than $M/2^{K-1}$ neural networks, the *mutation step* described above is not possible. Instead of that the attacker discards all tree parity machines which predicted less than U outputs τ^A in the last V learning steps, with $\tau^A = \tau^B$, successfully. In our simulations we use a limit of $U=10$ and a history of $V=20$ as default values. Ad-

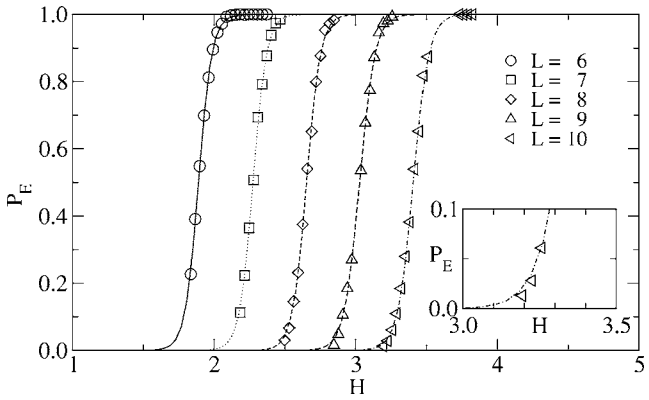

FIG. 4. Success probability P_E of the genetic attack for $K=3$, $N=1000$, random walk learning rule, and $M=4096$. Symbols represent results obtained from 1000 simulations, and the lines show a fit with Eq. (17).

ditionally, at least 20 neural networks are kept in such a *selection step*.

(iii) In the case of $\tau^A \neq \tau^B$ the attacker's networks remain unchanged, because A and B do not update the weights in their tree parity machines.

The attack is considered successful if at least one of E's neural networks has synchronized 98% of the weights before the end of the key exchange. We use this relaxed criterion in order to decrease the fluctuations of P_E [17].

The success probability of the genetic attack strongly depends on the value of the parameter H. This effect is clearly visible in Fig. 4. In order to determine P_E as a function of H, a Fermi-Dirac distribution

$$P_E = \frac{1}{1+\exp[-\beta(H-\mu)]} \tag{17}$$

with two parameters β and μ can be used as a fitting model. This equation is also valid for the geometric attack and the majority attack [14].

Figure 5 shows the results of the fits using Eq. (17). While

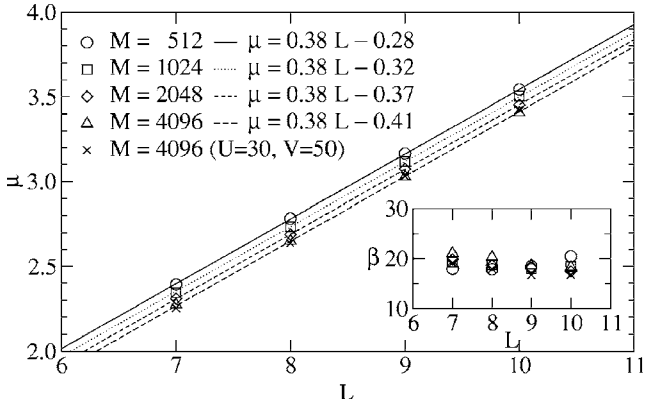

FIG. 5. Parameters μ and β as a function of the synaptic depth L. Symbols denote results of fitting simulation data for different M with Eq. (17) and the lines were calculated using the model given in Eq. (19).

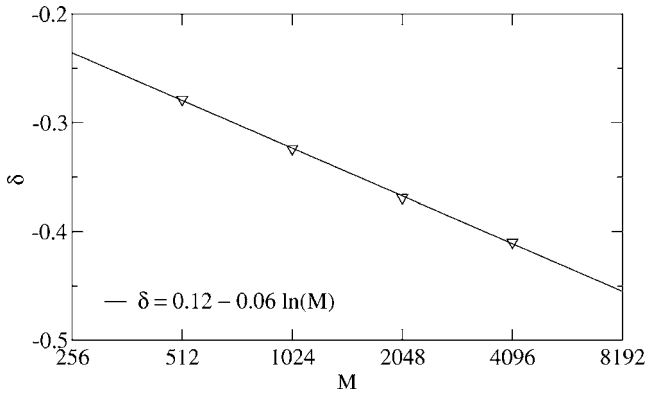

FIG. 6. Offset δ as a function of the number of attackers M, for $K=3$, $N=1000$, and the random walk learning rule. Symbols and the line were obtained by a fit with Eq. (19).

β is nearly independent of L and M, μ increases linearly with the synaptic depth,

$$\mu = \alpha_s L + \delta. \tag{18}$$

Obviously, the attacker can change the offset δ, but not α_s, by using more resources. As shown in Fig. 6 E needs to double M in order to decrease δ by a fixed amount $\gamma \ln 2$. Thus, μ is a linear function of both L and $\ln M$,

$$\mu = \alpha_s L - \gamma \ln M + \mu_0. \tag{19}$$

Substituting Eq. (19) into Eq. (17) leads to

$$P_E = \frac{1}{1+\exp[\beta(\mu_0 - \gamma \ln M)]\exp[\beta(\alpha_s - \alpha)L]} \tag{20}$$

for the success probability of the genetic attack as a function of $\alpha = H/L$, the synaptic depth L, and the maximal number of attackers M.

From these results we can deduce the scaling of P_E with regard to L and M. For large values of the synaptic depth the asymptotic behavior is given by

$$P_E \sim e^{-\beta(\mu_0 - \gamma \ln M)} e^{-\beta(\alpha_s - \alpha)L} \tag{21}$$

as long as $\alpha < \alpha_s$.

This equation shows that that the partners have a great advantage over an attacker. If A and B increase L, the success probability drops exponentially,

$$P_E \propto e^{-\beta(\alpha_s - \alpha)L}, \tag{22}$$

while the complexity of the synchronization rises only polynomially. This is clearly visible if one looks at the function $P_E(\langle t_{\text{sync}} \rangle)$, which is shown in Fig. 7. Due to the offset δ in Eq. (18) the attacker is successful for small values of L. But, for larger synaptic depth optimal security is reached for values of H and L, which lie on the envelope of $P_E(\langle t_{\text{sync}} \rangle)$. This curve is approximately given by $H = \alpha_c L$, as this condition maximizes $\alpha_s - \alpha$ while synchronization is still possible [14].

In contrast, the attacker has to increase the number of her tree parity machines exponentially,

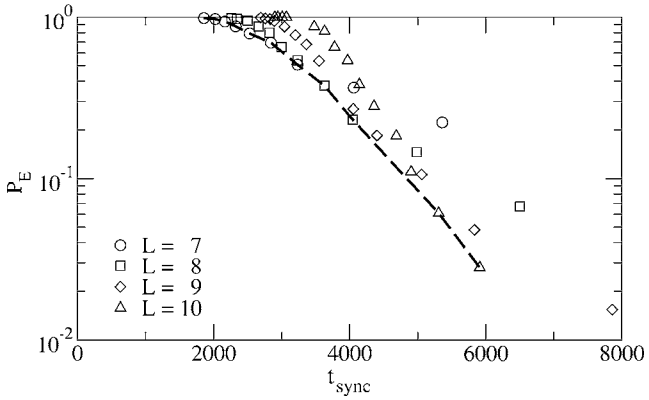

FIG. 7. Success probability of the genetic attack as a function of the synchronization time for $K=3$, $N=1000$, random walk learning rule, $M=4096$, and different values of L. The dashed line shows the envelope of this set of curves.

$$M \propto e^{[(\alpha_s-\alpha)/\gamma]L}, \tag{23}$$

in order to compensate a change of L and maintain a constant success probability P_E. But, this is usually not possible due to limited computer power.

Alternatively, the attacker could try to optimize the other two parameters of the genetic attack. As shown in Fig. 8, E obtains the best result if she uses $U=30$, $V=50$ instead of $U=10$, $V=20$. Figure 5 shows that this modification leads to a lower value of β, but does not influence $\mu(L)$. Therefore, E gains little, as the scaling relation (23) is not affected. That is why A and B can easily reach an arbitrary level of security.

IV. LEARNING RULES

Beside the random walk learning rule (7) used so far, there are two other suitable algorithms for updating the weights: the Hebbian learning rule (5) and the anti-Hebbian learning rule (6). The only difference between these three rules is whether and how the output σ_i of the hidden unit is included in the update step. But, this causes some effects which we discuss in this section.

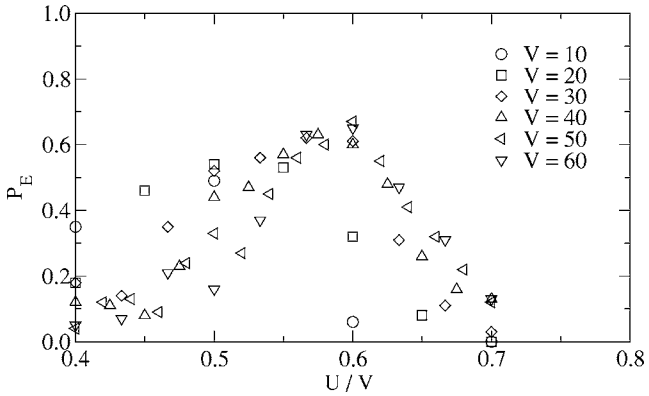

FIG. 8. Success probability of the genetic attack for $K=3$, $L=7$, $N=1000$, random walk learning rule, $M=4096$, and $H=2.28$. These results were obtained by averaging over 100 simulations.

In the case of the Hebbian rule A's and B's tree parity learn their own output. Therefore, the direction in which the weight w_{ij} moves is determined by the product $\sigma_i x_{ij}$. But, as the output of a hidden unit is a function of all input values, there are correlations between x_{ij} and σ_i. That is why the probability distribution of $\sigma_i x_{ij}$ is not uniformly distributed in the case of random inputs, but depends on the corresponding weight w_{ij},

$$P(\sigma_i x_{ij}=1)=\frac{1}{2}\left[1+\mathrm{erf}\left(\frac{w_{ij}}{\sqrt{NQ_i-w_{ij}^2}}\right)\right]. \tag{24}$$

According to this equation, $\sigma_i x_{ij}=\mathrm{sgn}(w_{ij})$ occurs more often than $\sigma_i x_{ij}=-\mathrm{sgn}(w_{ij})$. Thus, the Hebbian learning rule (5) pushes the weights towards the boundaries at $-L$ and $+L$.

In order to quantify this effect we calculate the stationary probability distribution of the weights. Using Eq. (24) for the transition probabilities leads to

$$P(w_{ij}=w)=p_0\prod_{m=1}^{|w|}\frac{1+\mathrm{erf}\left(\frac{m-1}{\sqrt{NQ_i-(m-1)^2}}\right)}{1-\mathrm{erf}\left(\frac{m}{\sqrt{NQ_i-m^2}}\right)}, \tag{25}$$

whereas the normalization constant p_0 is given by

$$p_0=\left(\sum_{w=-L}^{L}\prod_{m=1}^{|w|}\frac{1+\mathrm{erf}\left(\frac{m-1}{\sqrt{NQ_i-(m-1)^2}}\right)}{1-\mathrm{erf}\left(\frac{m}{\sqrt{NQ_i-m^2}}\right)}\right)^{-1}. \tag{26}$$

In the limit $N\to\infty$ the argument of the error function vanishes and the weights are uniformly distributed. In this case the synchronization process does not change the initial length

$$\sqrt{Q_i(t=0)}=\sqrt{\frac{L(L+1)}{3}} \tag{27}$$

of the weight vector.

But, for finite N the probability distribution (25) itself depends on the order parameter Q_i. Therefore, the expectation value of Q_i is the solution of the following equation:

$$Q_i=\sum_{w=-L}^{L}w^2P(w_{ij}=w). \tag{28}$$

By expanding Eq. (28) in terms of $N^{-1/2}$ we obtain

$$Q_i=\frac{L(L+1)}{3}+\frac{8L^4+16L^3-10L^2-18L+9}{15\sqrt{3\pi L(L+1)}}\frac{1}{\sqrt{N}}+O\left(\frac{L^4}{N}\right) \tag{29}$$

as a first-order approximation of Q_i for large system sizes. In the case of $1\ll L\ll\sqrt{N}$ the asymptotic behavior of this order parameter is given by

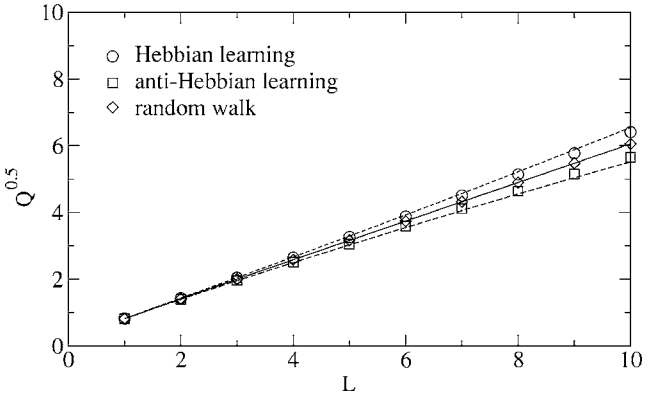

FIG. 9. Length of the weight vectors in the steady state for $K=3$ and $N=1000$. Symbols denote results averaged over 1000 simulations and lines show the first-order approximation given in Eq. (29) and Eq. (31).

$$Q_i \sim \frac{L(L+1)}{3}\left(1+\frac{8}{5\sqrt{3\pi}}\frac{L}{\sqrt{N}}\right). \tag{30}$$

Obviously, the application of the Hebbian learning rule increases the length of the weight vectors $\mathbf{w}_i$ until a steady state is reached. Additionally, the changed probability distribution of the weights affects the synchronization process and the success of attacks. That is why one encounters finite-size effects if $L/\sqrt{N}$ is large [17].

In the case of the anti-Hebbian rule A's and B's tree parity machines learn the opposite of their own outputs. Therefore, the weights are pulled away from the boundaries, so that

$$Q_i = \frac{L(L+1)}{3} - \frac{8L^4+16L^3-10L^2-18L+9}{15\sqrt{3\pi L(L+1)}}\frac{1}{\sqrt{N}} + O\left(\frac{L^4}{N}\right) \tag{31}$$

$$\sim \frac{L(L+1)}{3}\left(1-\frac{8}{5\sqrt{3\pi}}\frac{L}{\sqrt{N}}\right) \tag{32}$$

for $1 \ll L \ll \sqrt{N}$. Here, the length of the weight vectors $\mathbf{w}_i$ is decreased.

In contrast, the random walk learning rule always uses a fixed set output. Here, the weights stay uniformly distributed, because only the random input values x_{ij} determine the direction of the movements. In this case the length of the weight vectors is given by Eq. (27).

Figure 9 shows that the theoretical predictions are in good quantitative agreement with simulation results as long as L^2 is small compared to the system size N. The deviations for large L are caused by higher-order terms which are ignored in Eq. (29) and Eq. (31).

The choice of the learning rule affects synchronization with random inputs as well as with queries. As the prediction error (14) is a function of $h_i/\sqrt{Q_i}$, this ratio instead of just the local field determines the behavior of the system. That is why there are different values of α_c and α_s for each learning rule, which is shown in Fig. 10 and Fig. 11.

In the limit $N \to \infty$, however, a system using Hebbian or anti-Hebbian learning exhibits the same dynamics as observed in the case of the random walk rule for all system sizes. This is clearly visible in Fig. 11. Consequently, one can determine the properties of neural cryptography in the limit $N \to \infty$ without actually analyzing very large systems. It is sufficient to use the random walk learning rule and moderate values of N in simulations.

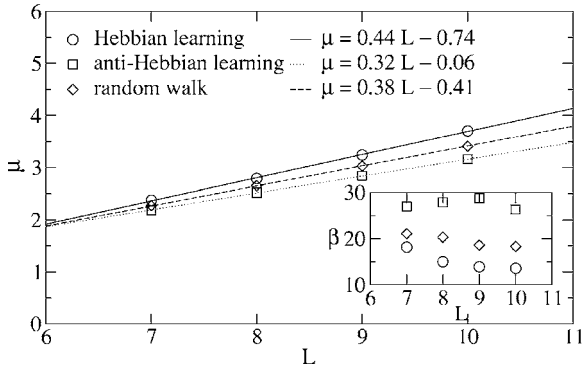

FIG. 10. Parameter μ and β as a function of L for the genetic attack with $K=3$, $N=1000$, and $M=4096$. The symbols represent results from 1000 simulations and the lines show a fit using the model given in Eq. (19).

V. SECURITY

In order to assess the security of the neural key-exchange protocol one has to consider all known attacks. Therefore, we compare the efficiency of several methods here.

Figure 12 shows that the success probability P_E drops exponentially with increasing synaptic depth L,

$$P_E \sim e^{-y(L-L_0)}, \tag{33}$$

as long as $L > L_0$. While this scaling behavior is the same for all attacks, the constants y and L_0 are different for each method.

The geometric attack is the simplest method considered here. E only needs one tree parity machine, but the success probability P_E is lower than for the advanced methods. As the exponent y is large, the two partners can easily secure the neural key-exchange protocol by increasing the synaptic depth [17].

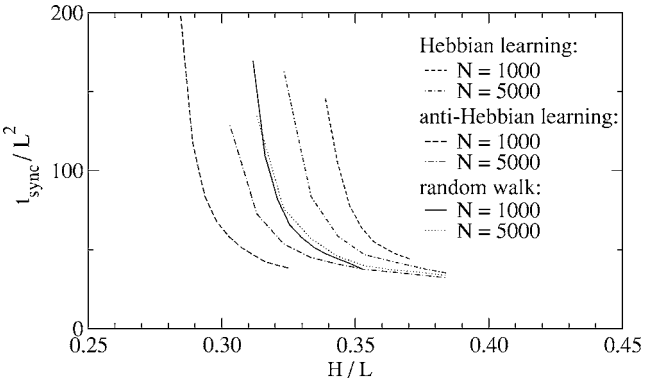

FIG. 11. Synchronization time t_{sync} as a function of H for $K=3$, $L=7$, averaged over 100 simulations.

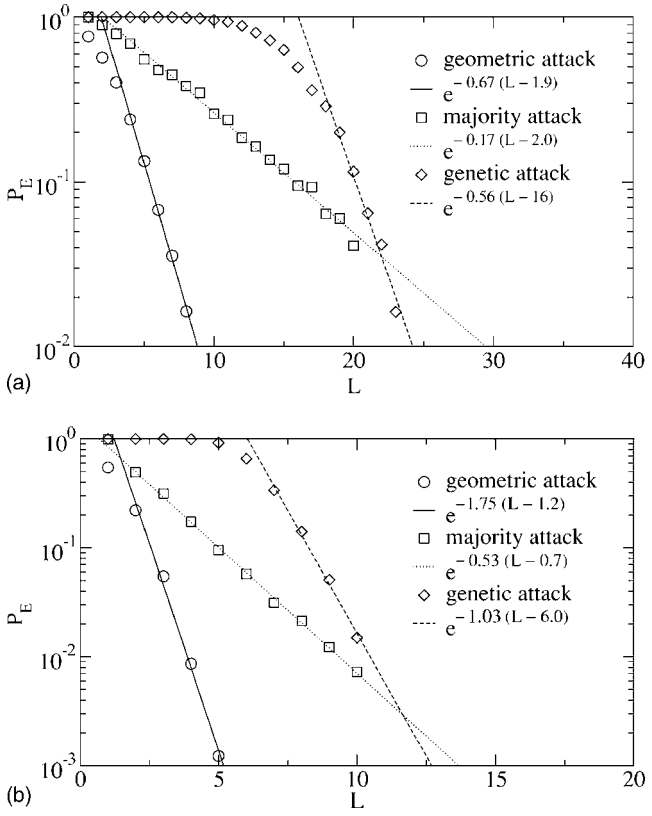

FIG. 12. Success probability P_E of the geometric attack with $M=1$, the majority attack with $M=100$, and the genetic attack with $M=4096$, for $K=3$, $N=1000$, and the random walk learning rule. Symbols represent results averaged over 1000 simulations, in part (a) for random inputs and in part (b) for queries with $H=0.32L$. The lines were obtained by fitting with Eq. (33).

In the case of the majority attack P_E is higher, because the cooperation between E's tree parity machines reduces the coefficient y. A and B have to compensate this by further stepping up L. In contrast, the genetic attack increases L_0, while y does not change significantly compared to the geometric attack. Therefore, the genetic algorithm is better only if L is not too large. Otherwise E gains most by using the majority attack.

As shown in Fig. 12 the partners can improve the security of the key-exchange protocol against all three attacks by using queries. However, the majority attack remains the most efficient of E's methods.

We note that these results are based on numerical extrapolations of the success probability P_E. While analytical evidence for the complexity of a successful attack would be desirable, it is not available yet in the case of the nondeterministic methods with $P_E<1$ discussed above. But, there are only two successful deterministic algorithms for E known at present: a brute-force attack or a genetic attack with $M=2^{(K-1)t_{\mathrm{sync}}}$ networks. The complexity of these attacks clearly grows exponentially with increasing L. Therefore, breaking the security of neural cryptography belongs to the complexity class NP (nondeterministic polynomial time), but we cannot prove that it is not in P (polynomial time). This situation is similar to that of other cryptographic protocols, e.g., the Diffie-Hellman key exchange [6].

VI. CONCLUSIONS

The security of cryptographic algorithms is usually based on different scaling laws regarding the computational complexity for users and attackers. By changing some parameter one can increase the cost of a successful attack exponentially, while the effort for the users increments only polynomially. For conventional cryptographic systems this parameter is the length of the key. In the case of neural cryptography it is the synaptic depth L of the neural networks.

As the neural key-exchange protocol uses tree parity machines, an attacker faces the challenge to guess the internal representation of these networks correctly. Learning alone is not sufficient to solve this problem. Otherwise the scaling laws hold and the partners can achieve any desired level of security by increasing L.

We have analyzed an attack, which combines learning with a genetic algorithm. We have found that this method is very successful as long as L is small. But, attackers have to increase the number of their neural networks exponentially in order to compensate higher values of L. That is why neural cryptography is secure against the genetic attack as well.

This method achieves the best success probability of all known methods only if the synaptic depth L is not too large. For higher values of L the attacker gains more by using the majority attack. But, both methods are unable to break the security of the neural key-exchange protocol in the limit $L\rightarrow\infty$.

Additionally, we have studied the influence of different learning rules on the neural key-exchange protocol. Hebbian and anti-Hebbian learning change the order parameter Q, which is related to the length of the weight vectors. If the system size N is small compared to L^2, this causes finite-size effects. But, in the limit $L/\sqrt{N}\rightarrow 0$ the behavior of all learning rules converges to that of the random walk rule.

Based on our results, we conclude that the neural key-exchange protocol is secure against all attacks known up to now. But—similar to other cryptographic algorithms—there is always a possibility that a clever method may be found which destroys the security of neural cryptography completely.

[1] I. Kanter, W. Kinzel, and E. Kanter, Europhys. Lett. **57**, 141 (2002).
[2] W. Kinzel and I. Kanter, J. Phys. A **36**, 11173 (2003).
[3] R. Metzler, W. Kinzel, and I. Kanter, Phys. Rev. E **62**, 2555 (2000).
[4] W. Kinzel, R. Metzler, and I. Kanter, J. Phys. A **33**, L141 (2000).
[5] A. Ruttor, W. Kinzel, L. Shacham, and I. Kanter, Phys. Rev. E

69, 046110 (2004).
[6] D. R. Stinson, *Cryptography: Theory and Practice* (CRC Press, Boca Ratan, FL, 1995).
[7] M. Volkmer and S. Wallner, in *Proceedings of the 2nd German Workshop on Mobile Ad-hoc Networks, WMAN 2004*, edited by P. Dadam and M. Reichert (Bonner Köllen Verlag, Ulm, 2004), Vol. P-50 of Lecture Notes in Informatics (LNI), pp. 128–137.
[8] M. Volkmer and S. Wallner, in *Proceedings of the 1st International Workshop on Secure and Ubiquitous Networks, SUN'05* (IEEE Computer Society, Copenhagen, 2005), pp. 241–245.
[9] M. Volkmer and S. Wallner, in *ECRYPT (European Network of Excellence for Cryptology) Workshop on RFID and Lightweight Crypto* (Graz University of Technology, Graz, 2005), pp. 102–113.
[10] M. Volkmer and S. Wallner, IEEE Trans. Comput. **54**, 421 (2005).
[11] A. Klimov, A. Mityaguine, and A. Shamir, in *Advances in Cryptology—ASIACRYPT 2002*, edited by Y. Zheng (Springer, Heidelberg, 2003), p. 288.
[12] L. N. Shacham, E. Klein, R. Mislovaty, I. Kanter, and W. Kinzel, Phys. Rev. E **69**, 066137 (2004).
[13] R. Mislovaty, E. Klein, I. Kanter, and W. Kinzel, Phys. Rev. Lett. **91**, 118701 (2003).
[14] A. Ruttor, W. Kinzel, and I. Kanter, J. Stat. Mech.: Theory Exp. (2005) P01009.
[15] I. Kanter and W. Kinzel, in *Proceedings of the XXII Solvay Conference on Physics on the Physics of Communication*, edited by I. Antoniou, V. A. Sadovnichy and H. Wather (World Scientific, Singapore, 2003), p. 631.
[16] W. Kinzel and I. Kanter, in *Advances in Solid State Physics*, edited by B. Kramer (Springer, Berlin, 2002), Vol. 42, pp. 383–391.
[17] R. Mislovaty, Y. Perchenok, I. Kanter, and W. Kinzel, Phys. Rev. E **66**, 066102 (2002).
[18] A. Ruttor, G. Reents, and W. Kinzel, J. Phys. A **37**, 8609 (2004).
[19] A. Engel and C. Van den Broeck, *Statistical Mechanics of Learning* (Cambridge University Press, Cambridge, 2001).
[20] L. Ein-Dor and I. Kanter, Phys. Rev. E **60**, 799 (1999).
[21] W. Kinzel and P. Rujan, Europhys. Lett. **13**, 473 (1990).