# A Key Establishment IP-Core for Ubiquitous Computing

Markus Volkmer and Sebastian Wallner
Hamburg University of Technology,
Department of Computer Engineering VI, D-21073 Hamburg, Germany
markus.volkmer@tuhh.de, wallner@tuhh.de

## Abstract

*A most critical and complex issue with regard to constrained devices in the ubiquitous and pervasive computing setting is secure key exchange. The restrictions motivate the investigation and discussion of alternative solutions. We suggest a low hardware-complexity solution for authenticated symmetric key exchange, using a Tree Parity Machine Rekeying Architecture. An authenticated key exchange is formulated from within the tree parity machine interaction concept and requires only few transmissions. It averts a Man-In-The-Middle attack and the currently known attacks on the non-numbertheoretic on principle. A key exchange can be performed within a few milliseconds, given typical limited bandwidth wireless communication channels. A flexible rekeying functionality enables the exploitation of the achievable key exchange rates. Characteristics of a standard-cell ASIC design realization as IP-core in* $0.18\mu$*-CMOS technology are evaluated.*

## 1. Introduction

In ubiquitous and pervasive computing scenarios, key exchange and entity authentication is of major importance with regard to security. Given a lack of infrastructure, approaches needing a central authority, like a trust center or another third trusted party securely providing public keys as e.g. in ID-based cryptosystems, are penalized.

Handheld devices, smartcards, mobiles or other wireless communication devices require security concepts to be developed, in order have privacy and (commercially) exploit the use of such devices [1]. An extreme example is given by the RFID-industry, that should have a particular interest because the secrecy of data is directly linked to the commercial prosperity of their products.

In ubiquitous and pervasive computing applications, such as sensor networks, RFID-systems or Near Field Communication (NFC), the devices in use as nodes of the network can impose severe size limitations and power consumption constraints. Consequently, the available size for additional cryptographic hardware components is also limited [14, 16]. Cryptographic methods with appropriate computational efficiency, that also consider a certain message or protocol overhead, are inevitable.

The exchange of a common secret key over a public channel is dominated by methods based on number theory such as RSA. Computational security is based on the difficulty of the discrete logarithm problem as in *El Gamal* [2], which is considered as difficult as the factorization problem of a product of long prime numbers as in *RSA* [12]. Such asymmetric algorithms need to perform a lot of computational intensive arithmetics on typically limited embedded microcontrollers in the ubiquitous and pervasive computing setting. Threshold cryptography is still computationally intensive and a distributed certificate authority does not address the resource limitations of devices.

The state-of-the-art, regarding applications in general embedded systems, is represented by *Elliptic Curve Cryptography* and the generalization to *Hyper-Elliptic Curves* (see e.g. [11]). Without a reduction of the security, these representations allow to reduce the size of the numbers to calculate with. Yet, more complex expressions need to be calculated. Often, a necessary tradeoff between the level of security and the available resources or computation time has to be faced. Seeking and investigating alternative approaches thus remains a challenge for research.

In this paper we suggest to discuss and investigate a hardware-friendly algorithm for secure symmetric key exchange by synchronization of so-called *Tree Parity Machines* [4]. We extend the proposed concept to an (entity) authenticated key exchange, that averts a Man-In-The-Middle attack and the currently known attacks. We employ and evaluate a small hardware solution presented in [15] also allowing for short key lifetimes with its flexible rekeying functionality. We focus on the low hardware-complexity of this IP-Core solution for secure data exchange between resource-limited devices.

## 2. Key Exchange by Tree Parity Machines

A symmetric key exchange method based on the fast synchronization of two identically structured interacting Tree Parity Machines (TPMs) was proposed by Kinzel and Kanter [4, 6]. The principle does not involve large numbers and principles from number theory, however, Shamir et al. conferred to this interaction over multiple rounds as *a gradual type of Diffie-Hellman* [7]. Even more related, secret key agreement based on interaction over a public insecure channel is discussed under information theoretic aspects by Maurer [8].

The particular tree structure has non-overlapping binary inputs, discrete weights and a single binary output as depicted in Fig. 1a. Their interaction protocol for key exchange is realized by a mutual adaptation process between the two parties $A$ and $B$, not involving large numbers and methods from number theory.

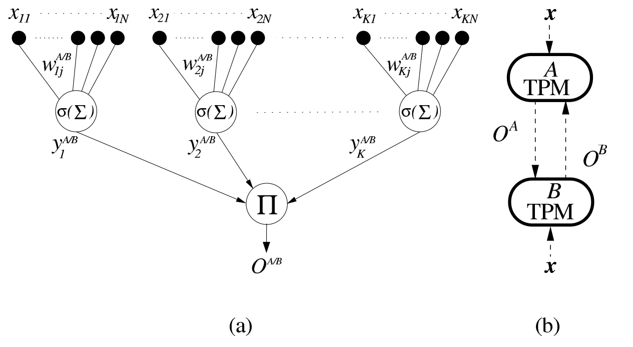


**Figure 1. (a) Tree Parity Machine (TPM). (b) Exchange of outputs for mutual learning.**

In the following, we describe the implemented parallel-weights version using hebbian learning (cf. [4, 7]). Weights are identical in both TPMs after synchronization. The notation $A/B$ denotes equivalent operations for the parties $A$ and $B$. A single $A$ or $B$ denotes an operation which is specific to one of the parties. The TPM (see Fig. 1a) consists of $K$ hidden units ($1 \leq k \leq K$) in a single hidden-layer with non-overlapping inputs (the tree structure) and a single unit in the output-layer. Each hidden unit receives different $N$ inputs ($1 \leq j \leq N$), leading to an input field of size $K \cdot N$. The vector-components are random variables with zero mean and unit variance. The output $O^{A/B}(t) \in \{-1, 1\}$, given bounded weights $w_{kj}^{A/B}(t) \in [-L, L] \subseteq \mathbb{Z}$ (from input unit $j$ to hidden unit $k$) and common random inputs $x_{kj}(t) \in \{-1, 1\}$, is calculated by a parity function of the signs of summations:

$$O^{A/B}(t) = \prod_{k=1}^{K} y_k^{A/B}(t) = \prod_{k=1}^{K} \sigma\left(\sum_{j=1}^{N} w_{kj}^{A/B}(t)\, x_{kj}(t)\right) . \quad (1)$$

The common random inputs can also be kept secret between the parties, yielding authentication (see Section 3). $\sigma$ is a party-specific modified sign-function, that defines an agreement between the two parties on an opposite sign in case of a sum $\alpha_k^{A/B}(t) \in \mathbb{Z}$ of zero.

The so-called *bit package* variant was chosen for implementation (cf. [4]). Due to an reduction of (physical) output exchanges by an order of magnitude down to around a dozen packages, it is advantageous for practical communication channels with a certain protocol overhead. Parties $A$ and $B$ start with an individual randomly generated initial weight vector $w_{kj}^{A/B}(t_0)$ – their secret. After a set of $b > 1$ presented inputs, where $b$ denotes the size of the bit package, the corresponding $b$ TPM outputs (bits) $O^{A/B}(t)$ are exchanged over the public channel in one package (see Fig. 1b). The $b$ sequences of hidden states $y_k^{A/B}(t) \in \{-1, 1\}$ are stored for the subsequent learning process. A hebbian learning rule is applied to adapt the weights, using the $b$ outputs and $b$ sequences of hidden states. They are changed only on an agreement $O^A(t) = O^B(t)$ on the parties' outputs. Furthermore, only weights of those hidden units are changed, that agree with this output ($O^{A/B}(t) = y_k^{A/B}(t)$):

$$w_{kj}^{A/B}(t) := w_{kj}^{A/B}(t-1) + O^{A/B}(t)\, x_{kj}(t) . \quad (2)$$

Updated weights are bound to stay in the maximum range $[-L, L] \subseteq \mathbb{Z}$ by reflection onto the boundary values.

In iterating the above procedure, each component of the weight vectors performs a random walk with reflecting boundaries. This implies a trajectory in a weight space of $(2L+1)^{KN}$ points. Two corresponding components in $w_{kj}^A(t)$ and $w_{kj}^B(t)$ receive the same random component of the common input vector $x_{kj}(t)$. After each bounding operation, the distance between the components is successively reduced to zero. Synchrony is achieved when both parties have learned to produce each other's outputs. They remain synchronized (see learning rule Eq. 2) and continue to produce the same outputs on every commonly given input. This effect in particular leads to common weight-vectors in both TPMs in each of the following iterations. These weights have never been communicated between the two parties and can be used as a common time-dependent key for encryption and decryption respectively.

A practical test for synchrony is defined by testing on successive equal outputs in a sufficiently large number of iterations $t_{min}$, such that equal outputs by chance are excluded. Our investigations/experiments confirmed that the average synchronization time is distributed and peaked around 400 for the parameters given in [4].

## 3. Security and Entity Authentication

For the key exchange protocol without entity authentication, eavesdropping attacks have concurrently been proposed by Shamir et al. [7] and Kanter, Kinzel et al. [10, 3, 5]. The prevalent definition of a successful attack is having an 98 percent average overlap with the coefficients of one party, when parties $A$ and $B$ are already synchronous and thus successfully finished the key exchange and the communication. The overlap $|w^E(t) \cdot w^{A/B}(t)|$ is averaged over all summation units. The authors chose this definition, because of the strong fluctuations observed in the success probability. The attacks in [7, 10, 3, 5] can all be made arbitrarily costly and thus can practically be defeated by simply increasing the parameter $L$. The security increases proportional to $L^2$ while the probability of a successful attack decreases exponentially with $L$ [10]. The approach is thus regarded computationally secure with respect to these attacks for sufficiently large $L$ [13, 5].

The latest attack, which does not seem to be affected by an increase of $L$ (but still by an increase of $K$) uses several coordinated and communicating TPMs [5]. A successful attack according to the definition given above could be achieved with a probability of 0.5. The success probability of achieving a 99 percent average overlap drops down to 0.25. However, an attacker does not know, which of the $K \cdot N$ components of the coefficients (the key) are correct. In currently used symmetric encryption algorithms, the flipping of a single bit only already leads to a complete failure in decryption. Due to the only partial knowledge of an attacker on the final key, an added or included privacy amplification through hashing can further significantly decrease this knowledge and increase the secrecy of the final key (compare e.g. [8]) and also the security of the trajectory mode. It increases the entropy of the keys and destroys partial knowledge an attacker might have gained on the key from the known attacks.

Note that all of the existing attacks refer to a non-authenticated key exchange, in which also Man-in-the-middle attacks are possible. Given an appropriate mechanism for entity authentication, e.g. as explained in the following, and the application of a privacy amplification step, an appropriate level of security is provided at least for some applications in embedded security.

The structure of the network, the involved computations producing the output $O^{A/B}(t)$ (Eq. 1), the adaptation-rule (Eq. 2) and especially the common inputs $x_{kj}(t)$ are public in the original formulation. The only secrets involved are the different initial weights $w_{kj}^{A/B}(t_0)$ of the two parties. If they were not secret, the resulting keys could simply be calculated (by an adversary), because all further computations are completely deterministic. An extension to include authentication into the key exchange protocol is derived from the observation, that two parties $A$ and $B$ which do not have the same input vectors

$$\forall t: \; x^A(t) \neq x^B(t) \tag{3}$$

cannot synchronize. Note, that the aim of the two-party-system is to learn each other's outputs on commonly given inputs. Lacking common inputs, the two parties are trying to learn completely different mappings between inputs $x^{A/B}(t)$ and outputs $O^{A/B}(t)$. Consequently, the two parties cannot synchronize, there will not be time-dependent equal weights $w^{A/B}(t)$ and thus no exchange of a key. This again is exactly the service one would want to restrict only to authorized parties by employing any other explicit authentication using some common information.

For demonstration we consider the distance $d$ at time $t$ between two weight-vectors as their normalized sum of absolute differences. We observe the development of distance $d(w^A(t), w^B(t)) \in [0, 1]$ over time for different offsets ($\forall t: \; x^A(t) = x^B(t + \Delta), \; \Delta \in \mathbb{N}$) in the (pseudo-random) input-sequence and for completely different input-sequences. Fig. 2 depicts, that the distance between two parties that do not possess the same inputs remains fluctuating and never decreases towards zero. We also investigated different offsets with the same qualitative outcome. Two parties with completely different inputs (although not realistic given a concrete and publicly known Linear Feedback Shift Registers (LFSR) as pseudo random number generator) show the same qualitative behavior. Two parties having the same inputs (offset zero) soon decrease their distance and synchronize.

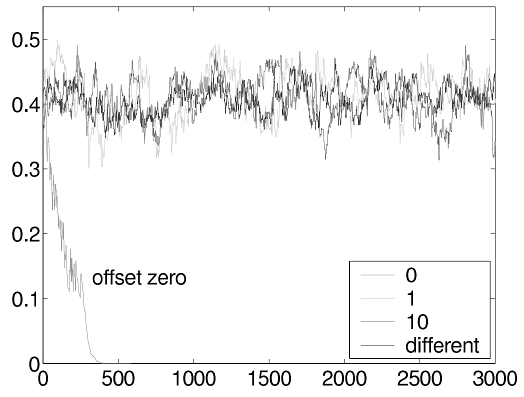


**Figure 2. Distance $d$ vs. the number of iterations $t$ for different offsets in the inputs.**

The random walks with reflecting boundaries performed by the weights in the iterative process now make uncorrelated moves and moves in the wrong direction. Two corresponding components $w_{kj}^A(t)$ and $w_{kj}^B(t)$ now receive a different random component $x_{kj}(t)$ of their (differing) input vectors (cf. Eq. 1). The distance between the components is thus no longer successively reduced to zero after each bounding operation and the two parties diverge.

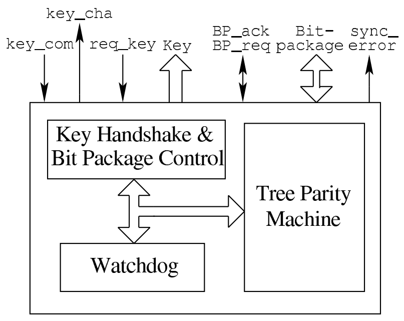
The non-synchronization in the case of no common inputs, therefore enables us to incorporate authentication by keeping the common (pseudo-random) inputs $x^{A/B}(t)$ secret between the two parties in addition to their individual secret (random) initial weights $w^{A/B}(t_0)$. There are $2^{KN}-1$ possible common inputs as second initial secrets, which is a large enough practical amount for the parameters as chosen in [4] that makes brute force attacks computationally very expensive. Even more, a Man-In-The-Middle attack and the currently known attacks [7, 6] all using TPMs are averted on principal by this entity authentication, because it suppresses the necessary synchronization. It is important to note, that such a second secret in this symmetric principle does not represent any principal disadvantage, because a basic common information is always also necessary in other authentication protocols (*cf.* [9]) and is already necessary to avert a Man-In-The-Middle attack alone.

## 4. TPM Rekeying Architectures

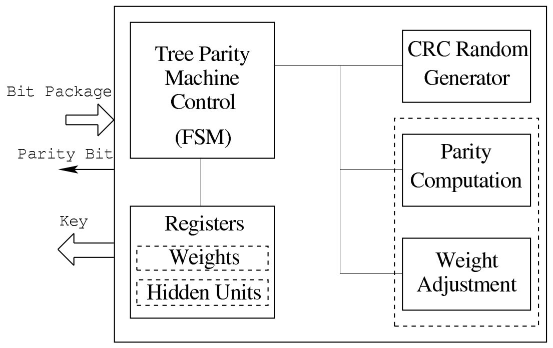
With respect to a hardware implementation (see [15] for more details), note that only signs and bounded integers are processed within the algorithm. The result of the outer product in Eq. 1 can be realized without multiplication. The product within the sum is only changing the sign of the weight. Thus, the most complex structure to be implemented is an adder. The branches are only based on a test for the sign or a test on equality to zero, also easily done in hardware.

Furthermore, only sign-operations and additions are present in the learning rule (Eq. 2), well suited for a hardware implementation. The amount of registers needed for storage increases in the bit package variant, finally imposing a tradeoff area vs. speed. Equal (pseudo-)random inputs are realized by equally initialized LFSR or a Cyclic Redundancy Code (CRC). Different (secret) initial weights can either be fixed (device-specific), or they can be provided by an additional application-specific device or by a thermal noise device. The synchronization criterion basically comprises a counter.

The Tree Parity Machine Rekeying Architectures (TPMRAs) [15] are functionally separated into two main structures. One structure essentially comprises the Key Handshake and Bit Package Control. The other structure contains the TPM Unit and its control state machine. As described in Section 2, we implemented the *bit package* generalization of the protocol (cf. [4]). The overall structure of the TPMRAs is shown in Fig. 3a. It consists of three functional blocks: a Key Handshake and Bit Package Control, the TPM



(a)



(b)

**Figure 3. (a) Basic TPM Rekeying Architecture. (b) Internal structure of the TPM Unit.**

unit and a Watchdog timer. The Watchdog timer supervises the number of interactions needed for a key-exchange between two parties. If there is no synchronization within a specific time (remember that the synchronization time is distributed), a signal (`sync_error`) indicates a synchronization error. It is programmable for variable average synchronization times subject to the chosen TPM structure.

The Key Handshake and Bit Package Control handles the key transmission with an encryption unit and the bit package exchange process with the other party. It accomplishes the bit packaging by partitioning the parity bits from the TPM unit in tighter bit slices. Due to different computation cycles between two key exchange parties, the rekeying procedure employs a key request (`req_key`), a key changed (`key_cha`) and a key commit (`key_com`) handshake protocol (see Fig. 3a). A key is handed over via the internal bus (`Key`) to an encryption unit when the synchronization process is finished. For our application domain we choose a fixed bit package length of 32 bit

(`Bit Package`). The bit package exchange process uses a simple request/acknowledge handshake protocol (`BP_ack`, `BP_req`).

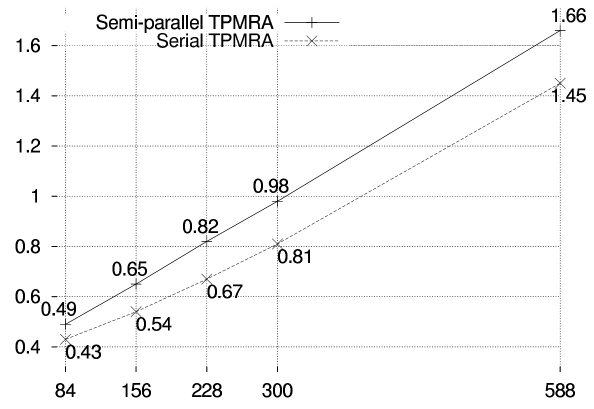
The TPM unit comprises the logic for the TPM structure, such as the logic for calculating the parity bits as explained in Section 2. It consists of the TPM control, a CRC-generator, a Parity Computation unit and a Weight Adjustment unit. A register bank holds the data for the hidden unit and the weights of the network as shown in Fig. 3b. The TPM control is realized as simple finite state machine (FSM) which executes the initialization of the TPM and the learning process with the bit package from the other party. The Parity Computation unit calculates the summation and the parity bit (Eq. 1). The weight adjustment unit accomplishes the learning rule (Eq. 2). The CRC random generator generates the pseudo random bits for the inputs of the TPM. It is initialized by a vector which is equal for both parties. For the purpose of entity authentication, only the initial value would have to be kept secret.

## 5. Results and Evaluation

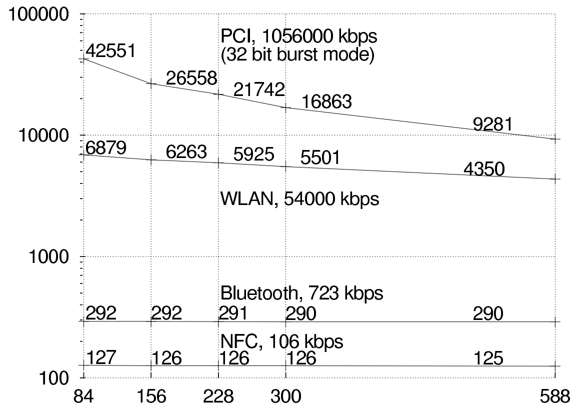
We employ parameterizable serial and semi-parallel TPM Rekeying Architectures as designed in [15]. In the serial architecture, the hidden-unit summation is performed by Time Devision Multiple Access (TDMA) of an $L$-bit adder, while the semi-parallel form uses TDMA of six $L$-bit adders in parallel. A standard cell ASIC prototype realization was build to verify the suitability of the TPMRA as IP-Core in devices with limited resources in the ubiquitous and pervasive computing setting. The underlying process was a $0.18\mu$ six-layer CMOS process with $1.8V$ supply voltage based on the *UMC library*.

The linear complexity of the protocol scales with the size $K{\cdot}N$ of the TPM structure, which defines the size $K \cdot N \cdot L$ of the key. We chose $K = 3$, a maximal $N = 49$ and $L = 4$, leading to a key size of up to 588 bit. The details of the TPMRA implementations (key length, serial or semi/fully-parallel realization) must be chosen with respect to the target environment, including the used parameters, the timing, the available channel capacity and the available chip-area, of course. As already investigated in [15], the area (Fig. 4a) of the TPMRA realizations scales approximately linear due to the linear complexity of the adders and ranges around one square-millimeter for the investigated key sizes. Note, that most of the area is consumed by the bit packaging, because of the necessary storage of the inputs for the adaption (see Section 4).

Here we additionally established the throughput (i.e. keys per second) for the serial TPMRA subject to the average synchronization time of 400 iterations in Fig. 4b. Four real communication channels and their bandwidths were chosen and the chosen log-scale allows to still see the small difference regarding the throughput for Bluetooth and NFC in comparison to WLAN and PCI (for comparison). For every protocol, we used the minimum available packet length due to our bit packages of 32 bit: (PCI 32 bit burst mode), WLAN 801.11g 512 bit, Bluetooth 190 bit and a Near Field Communication channel 64 bit. A comparison among the different communication channels indicates different slopes of the calculated (approximately linear) throughput characteristics. They denote the rising influence of the bit packaging calculations at smaller key lengths for channels of higher bandwidth such as WLAN (or PCI).



(a) Chip-area[$mm^2$] vs. key length[$bit$]



(b) Average key exchange rate[$Hz$] vs. key length[$bit$]

**Figure 4. Post-synthesis results for chip-area (logic) vs. key length (a) and (b) average key exchange rate vs. key length.**

Thus, the slope of the WLAN throughput characteristic is significantly higher than for Bluetooth and NFC. As expected, the influence of the channel bandwidth significantly determines the performance of the key exchange protocol. Obviously, the bottleneck is the underlying communication-bus. Given a high-speed communication channel, the proposed key exchange and rekeying in the *kHz*-range allows us to use rather weak encryption algorithms (cf. Section 3), as the security may rely on fast rekeying and the short key lifetimes. Of course, sophisticated encryption algorithms like AES or 3-DES can also be used.

## 6. Summary and Outlook

In the context of ubiquitous and pervasive computing and its special restrictions we promote the discussion of alternative security solutions. A method for authenticated symmetric key exchange for devices with severely limited resources based on Tree Parity Machine Rekeying Architectures [15] was suggested.

The silicon area of the used architectures lie within a square-millimeter and allow to exchange keys of practical sizes within milliseconds. The proposed exchange also allows for efficient rekeying schemes and short key lifetimes. The authenticated exchange of one 588 bit key on average demands to transmit 400 bit in 13 Bit Packages of 32 bit each. An implementation in general embedded systems can be done with only small overhead. Thus we suggest it as an IP-core for (wireless) computing devices of limited resources and even more in moderate security scenarios. The currently known attacks are successful only with a certain probability and all refer to a non-authenticated key exchange. The proposed entity authentication defeats these attacks on principle.

A future fully serial realization of the architecture will use TDMA of a single hidden unit. This further decreases the silicon area consumption but at the cost of an increase in necessary cycles for one output bit. This is a favorable design given low-bandwidth communication channels. Secure group communication in a broadcast or multicast and thus key exchange between multiple parties is also possible using several strategies currently under investigation. An efficient authentication and key management is needed here, due to frequent key exchanges on join or leave actions. The integration into concrete devices and its practical evaluation is also subject to future work.

## References

[1] R. Anderson. Protecting embedded systems – the next ten years (invited talk). In *Proc. of the 3rd International Workshop on Cryptographic Hardware and Embedded Systems, CHES 2001*, volume 2162 of *LNCS*, pages 1–2, Paris, France, May 14-16, 2001. Springer Verlag.

[2] T. E. Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, IT-31(4):469–472, 1985.

[3] I. Kanter and W. Kinzel. Neural cryptography. In *Proc. of the 9th International Conference on Neural Information Processing*, Singapore, Nov. 2002.

[4] I. Kanter, W. Kinzel, and E. Kanter. Secure exchange of information by synchronization of neural networks. *Europhysics Letters*, 57(1):141–147, 2002.

[5] I. Kanter, W. Kinzel, L. Shacham, E. Klein, and R. Mislovaty. Cooperrating attackers in neural cryptography. *Phys Rev. E*, 69(066137), 2004.

[6] E. Klein, R. Mislovaty, I. Kanter, A. Ruttor, and W. Kinzel. Synchronization of neural networks by mutual learning and its application to cryptography. In *Proc. of Neural Information Processing Systems 2004 (NIPS 2004)*, Whistler, British Columbia, Canada, Dec. 14-16 2004.

[7] A. Klimov, A. Mityagin, and A. Shamir. Analysis of neural cryptography. In *Proc. of AsiaCrypt 2002*, volume 2501 of *LNCS*, pages 288–298, Queenstown, New Zealand, December 1-5 2002. Springer Verlag.

[8] U. Maurer. Secret key agreement by public discussion. *IEEE Transactions on Information Theory*, 39(3):733–742, 1993.

[9] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 5th edition, 2001.

[10] R. Mislovaty, Y. Perchenok, I. Kanter, and W. Kinzel. Secure key-exchange protocol with an absence of injective functions. *Phys. Rev. E*, 66(066102), 2002.

[11] J. Pelzl, T. Wollinger, and C. Paar. Low cost security: Explicit formulae for genus-4 hyperelliptic curves. In *10th Annual Workshop on Selected Areas in Cryptography (SAC 2003)*. Springer Verlag, 2003.

[12] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public key cryptosystems. *Communications of the ACM*, 21(2):120–126, feb 1978.

[13] M. Rosen-Zvi, E. Klein, I. Kanter, and W. Kinzel. Mutual learning in a tree parity machine and its application to cryptography. *Phys. Rev. E.*, 66(066135), 2002.

[14] F. Stajano. Security in pervasive computing. In *Proc. of the 1st International Conference on Security in Pervasive Computing (SPC 2003)*, volume 2802 of *LNCS*, page 1. Springer Verlag, 2003.

[15] M. Volkmer and S. Wallner. Tree parity machine rekeying architectures. *IEEE Transactions on Computers*, 54(4):421–427, Apr. 2005.

[16] T. Wollinger, J. Guajardo, and C. Paar. Cryptography in embedded systems: An overview. In *Proc. of the Embedded World 2003 Exhibition and Conference*, pages 735–744, Nürnberg, Germany, Feb. 18-20 2003. Design & Elektronik, Nürnberg.