Side-Channel Resistance Evaluation of a Neural Network Based Lightweight Cryptography Scheme

Marc Stöttinger, Sorin A. Huss
Integrated Circuits and Systems Lab
Dept. of Computer Science
Technische Universität Darmstadt
Email: {stoettinger|huss}@iss.tu-darmstadt.de

Sascha Mühlbach
Research Group on Secure Things
Center of Advanced Security
Research Darmstadt (CASED)
Email: sascha.muehlbach@cased.de

Andreas Koch
Embedded Systems and Applications
Dept. of Computer Science
Technische Universität Darmstadt
Email: koch@esa.cs.tu-darmstadt.de

***Abstract*— Side-channel attacks have changed the design of secure cryptographic systems dramatically. Several published attacks on implementations of well known algorithms such as, e.g., AES, show the need to consider these aspects to build more resistant cryptographic systems. On the other hand, with the increasing use of cryptography in embedded systems a significant demand exists for cryptographic algorithms that are both resource- and power-efficient. These can be either modified existing or completely new ones. One of the candidates for such a new algorithm is the Tree Parity Machine Public Key Exchange, an algorithm based on artificial neural networks. While it has been evaluated in a number of practical applications in the past, its side-channel resistance has not been examined yet. We would like to close this gap and present a side-channel attack strategy as well as results gathered from measurements made on a real implementation.**

I. Introduction and Related Work

Security and integrity aspects play an important part in the design of current embedded systems. Identification cards, gaming consoles, SIM cards or Digital Rights Management systems for audio and video content require a strong security concept to protect data from unauthorized access, duplication or forgery. As hardware resources in these devices are often limited, intensive research aims to find optimized implementations of traditional algorithms (e.g., RSA, AES) or completely new algorithms with low hardware resource costs (sometimes also referred to as "'lightweight"' implementations/algorithms) [1]. On the opposing side, new techniques for attacking the implementations of cryptographic algorithms have been discovered. In recent years, aided by increasingly accurate measurement equipment, especially side-channel attacks which attempt to exploit information leaking from a device while it is performing cryptographic operations have been published [2]–[5]. Algorithms / implementations must thus be designed not only for computational efficiency, but also for a resistance against well-known side-channel attacks.

To this end, we will examine a recently published algorithm for public key exchange targeted especially for the use in resource-constrained environments. In contrast to the currently dominant algorithms, the proposed cryptographic system does not rely on number theory and complex mathematical calculations. Instead, its security is based on the synchronization of special neural networks by mutually adapting their internal states [6], an operation which can be implemented with very low hardware requirements. By using an appropriate learning rule, these tree-structured neural networks (called Tree Parity Machines, TPM) will synchronize to a common state when they are trained to imitate the output of the corresponding network on a set of common inputs. Since the internal state is never transmitted over the insecure channel, it can be used as a common encryption / decryption key for, e.g., an AES algorithm after synchronization has completed. Synchronization time is short and only a few hundred bits need to be transmitted to securely exchange a 128 bit symmetric key [7]. Furthermore, the computational complexity of the operations performed by the cryptographic devices is very low.

The security of the TPM algorithm has first been evaluated by Shamir et. al [8], discovering some weaknesses that could be exploited by a group of cooperative attackers. In recent years, however, a number of publications presented countermeasures alleviating these weaknesses (e.g., by adjusting the network parameters [7], or by adding predictable errors to the network output to confuse the attacker [9]). With these improvements, TPM has become a promising algorithm for use in resource-constrained environments.

Beyond the theroretical analysis, a number of case studies have examined the practical use of TPMs: [10] presents an architecture for secure chip-to-chip communication in embedded systems, using TPM key exchange extended with multiparty functionality for an unlimited number of bus participants. The same authors proposed a special stream cipher based on TPMs, which allows high-speed encryption and decryption at native bus speeds with very low resource demands [11]. The technique was also used for One-Time Password schemes [12], secure authentication in WiMAX networks [13], and supporting secure group communication in ad-hoc networks [14].

However, all the contributions did not discuss the side-channel resistance of implementations of the TPM algorithm (as demanded by [15]). We close this gap by providing a first evaluation of the side-channel resistance of the TPM algorithm and offer practical results gained by attacking an actual hardware implementation of the algorithm using the well-studied Differential Power Analysis (DPA) method.

The paper is organized as follows: Section II describes the

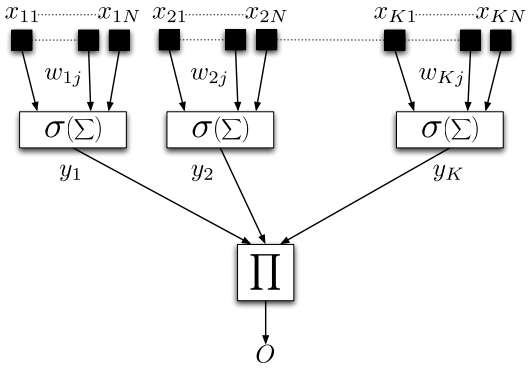

Fig. 1. Tree parity machine scheme

TPM-based cryptographic algorithm, while Section III covers implementation details. Section IV explains strategies to attack the TPM-based system by a side-channel analysis, followed by a discussion of the results of the practical experiments in Section V. We close with a conclusion and an outlook towards further research in the last Section.

II. Tree Parity Machine Key Exchange

For simplicity, we assume to have just two nodes A and B participating in the key exchange, but the approach is easily extensible to more nodes [14]. The TPM of a node is a network structure with K parallel hidden units ($1 \leq k \leq K$) and a single unit in the output-layer, arranged in a tree structure (see Fig. 1). Each hidden unit has $1 \leq j \leq N$ integer weights $w_{kj} \in [-L, L]$ and the same number of inputs x_{kj}. For brevity, the weights and inputs of a hidden unit k will be referred to as the vectors $\bar{w}_k$ and $\bar{x}_k$ later. Every input has a connection to exactly one weight. The inputs are streams of time-dependent random numbers $x_{kj}(t) \in \{-1, 1\}$ applied to both nodes in parallel. The initial weight values should be random and they must be kept private to each node for the cryptographic system to be secure. For each time step, the network output $O(t) \in \{-1, 1\}$ of each node is calculated by a parity function of the signs $y_k(t) \in \{-1, 1\}$ of sums:

$$O(t) = \prod_{k=1}^{K} y_k(t) = \prod_{k=1}^{K} \sigma\left(\sum_{j=1}^{N} w_{kj}(t)\ x_{kj}(t)\right) \qquad (1)$$

$\sigma(\cdot)$ computes the sign of its argument as 1 or -1. However, it has a slightly different definition for the separate nodes: For the node *initiating* the key exchange $\sigma(0) = 1$, for the responding node $\sigma(0) = -1$ holds.

A calculated output value is then transmitted over the (potentially insecure) channel to the other node. The nodes then perform an adaption step using the rule in Eq. (2).

$$w_{kj}(t) := \begin{cases} w_{kj}(t-1) + O(t)\ x_{kj}(t) & , O_A(t) = O_B(t) \wedge \\ & \quad O(t)\ y_k(t) < 0 \\ w_{kj}(t-1) & , \text{otherwise} \end{cases} \qquad (2)$$

This rule reinforces matching node outputs O for the parties. It has been shown to lead to faster convergence than rules aiming to attenuate mismatched node outputs [7] (which are simply skipped here). When node outputs match, the second condition limits adaptation to just those hidden units which *disagree* with the node output. Furthermore, the weights are clamped to always be inside the interval $[-L, L]$.

The steps of output calculation, output transmission, and network adaption are repeated multiple times. By using the same set of inputs $x_{kj}(t)$ for both nodes, the weight-vectors successively converge to each other [6]. When they become equal, both parties produce the same outputs. The adaptation rule ensures that further value changes now occur in lock-step at both nodes (remember that Eq. (2) leads to adaptation on *matching* outputs). We thus generate a sequence of time-dependent weight-vectors common to both TPMs.

As these weights (represented by bit vectors in an actual implementation) have not been exchanged over the insecure channel, there must be an indirect termination condition which detects the end of the key exchange process. This could be, e.g., realized by regular transmissions of a hash checksum derived from the current weight state, but on the other hand such an approach would lead to additional potential leakage and additional resource demands. In practice, experiments have shown [7] that observing a sequence of matching outputs for sufficiently long can be used to establish whether the two nodes have become synchronized with only a negligibly low risk of falsely assumed synchronization on chance matches. This is feasible because after nodes have synchronized at all, the TPMs continue to run in lock-step (producing the same outputs) and will never lose synchronization ever again. Since the weights are node-private, they can from then on be used as a symmetric key for encryption and decryption, respectively.

Synchronization time has been experimentally observed in [7] to be finite for integer weights and to peak at around 400 transmitted bits for practical network configurations ($K = 3 \ldots 15$, $N = 3 \ldots 21$, $L = 3$), leading to key sizes between 128 and 512 bits. Actual performance can be improved further by using the so-called *bit-package* variant [7], which speeds-up the synchronization process by reducing the communication overhead. Assuming that we send packets of 32 bits in parallel (which is feasible for embedded devices), synchronization is achievable in just $12 \ldots 14$ transmitted packets [11].

Since an attacker can observe both inputs $\bar{x}_k(t)$ and outputs $O(t)$ of the nodes, the security of the TPM key exchange relies on the initial weights being private. Otherwise, assuming that the attacker had full knowledge of the internal design of the nodes (K, N, etc.), he could just perform the same computations as the communicating nodes and also end-up being synchronized and able to follow the conversation. In some cases, it is actually possible to increase security further by also keeping $\bar{x}_k(t)$ secret [16]. This provides a possibility for implicit authentification of two or more parties due to the fact that synchronization of nodes is solely possible when the inputs are common.

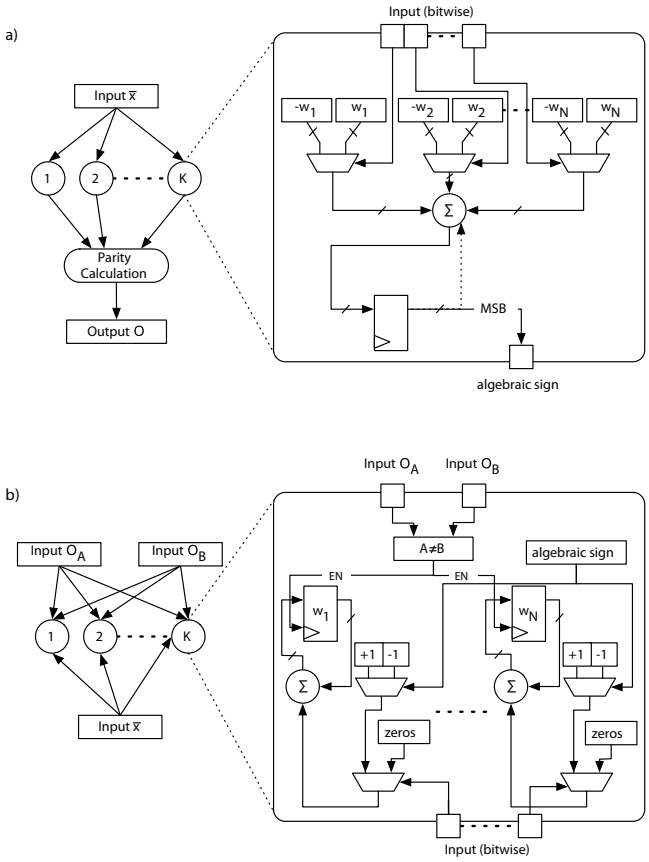

Fig. 2. Implementation of a) parity calculation and b) adaptation

III. Architectural Details of a TPM Implementation

Implementations of the Tree Parity Machine key exchange are often partitioned (see [7] and [10]) into parity calculation and weight adaption blocks (see Fig. 2). For calculating the parity output, the weight and input values of each hidden unit are loaded from a register file, multiplied (which only alters the sign bit) and then fed into an adder. Next, the sign bit of the resulting sum in every hidden unit is extracted and stored (for later use in the adaptation step). The stored sign bits are then used as inputs for the parity calculation. In bit packet mode, this step is repeated multiple times, thus resulting in a bit vector holding the parity output bits.

In practice, the data stream of input values is generated by using a linear feedback shift register (LFSR). It is initialized using a common value (which can be fixed, either public or hidden, or dynamically exchanged over the insecure channel), and then provides a reproducible sequence of bits. This, the input bits $\bar{x}_k(t)$ do not actually have to be exchanged between nodes, they are generated by the node-local LFSRs. This allows the low-bandwidth transmissions described in the previous Section.

For the adaptation step, the outputs of both nodes are compared with the previously stored signs of summation of every hidden unit. Depending on being equal or different, the weight values of the corresponding hidden unit are altered by +1 or -1 (respectively).

Parallelism can be exploited during both operations. In both parity and adaptation, hidden units are computationally independent (except for the final reduction step in the parity calculation). Given enough hardware area parity and adaptation computations could be performed in a single clock cycle each.

In practice, the achievable degree of parallelism is limited mainly by the required parallel adder for the sum calculation (large parallel adders are relatively large and slow) in every hidden unit. As the TPM is targeted at embedded devices with more restricted hardware resources, real implementations trade-off parallelism and resource demands (see [10]) to achieve good performance rates at acceptable costs. A realistic system could, e.g., process just three weights during parity calculation and just six weights during adaptation in parallel in every cycle.

Note that register accesses (characterized according to direction and frequency) during the computations are of great interest for side-channel analysis attacks, we will thus consider them in more detail in the next Section.

IV. Potential Side-channel Leakages of the TPM Implementation

Since Kocher et al. [17] introduced the side-channel analysis in 1999, published research concentrated on attacks on well-know block ciphers DES and AES. [2]–[5]. Dynamic power consumption is the most popular side-channel used for attacks, spawning the discipline of differential power analysis (DPA). It exploits the information leaked in the data dependent power draw of the target device. In this Section we discuss applying DPA to our TPM-based key exchange.

The effectiveness of DPA is highly dependent on the power model employed. In contrast to AES and DES, the TPM implementation under attack does not include any bijective non-linear function with a significant hamming distance between different intermediate values, e.g., a S-Box. It thus becomes more difficult to relate clearly distinguishable power consumption to these intermediate values (from which the attack aims to derive the private information).

As stated in Section II, the only private information in the TPM public key exchange scenario is the initial value of the weight vector in every node. Therefore, getting information allowing the derivation of this value will be the target of our attack.

As shown in Fig. 2a), the summation during parity calculation operates directly on the node weights. Together with the publically known input stream $\bar{x}_k(t)$, we will subject it to a side-channel analysis by DPA. We have to prevent the adaptation of weights during our attack and will thus send a constant $O_a = O_b$ to both nodes.

For this scenario, a Hamming weight power model (HW) of the summation step, shown for one hidden node of the TPM in Eq. 3, appears to be suitable. We interpret $\bar{x}(t_0)$ as plain text (at time t_0) and $\bar{w}$ as secret key. Our hypothesis estimates the

changes at the output of the summation of weights to derive the power consumption for each summation.

$$\mathcal{P}_{HW}(\bar{x}(t_0), \bar{w}) = HW\left(\sum_{j=1}^{N} w_j \; x_j(t_0)\right) \quad \text{for all } \bar{w} \in [-L, L]^N, \text{ with } \bar{x}(t_0) \in [-1, 1]^N \tag{3}$$

To generate estimated power sequences for all weight vectors, we assume the attacker knows *range* of weights values $[-L, L]$. Since all terms summed in Eq. (1) can be computed independently for each node, it is sufficient to examine only a single node in Eq. 3. Furthermore, the computation in Eq. 3 is time-independent (as opposed to the summed term in Eq. 1), since we prevented adaptation from altering the weights.

We can improve upon HW by analyzing the data-dependent power consumption leakage with respect to the *previous* value of the summation register (see Fig. 2a). This leads to a Hamming *distance* model (HD, Eq. 4) which now considers not only $\bar{x}(t_0)$ at time t_0, but also the value of the previous input $\bar{x}(t_{-1})$ at t_{-1} to estimate the power consumption. The power drawn during the computation depends on the number of flipped bits in the summation register, which can be more accurately predicted by also considering the previous sum. This, in turn, allows a more precise power estimation than the HW model.

$$\mathcal{P}_{HD}(\bar{x}(t_0), \bar{x}(t_{-1}), \bar{w}) = \mathcal{P}_{HW}(\bar{x}, \bar{w}) \oplus \mathcal{P}_{HW}(\bar{x}(t_{-1}), \bar{w}) \quad \text{for all } \bar{w} \in [-L, L]^N, \text{ with } \bar{x}(t), \bar{x}(t-1) \in [-1, 1]^N \tag{4}$$

V. Discussion of Experimental Results

We now discuss the results of several experiments analyzing the leakage of the TPM-based key exchange. They were carried out on the SASEBO platform [18], an FPGA prototyping board dedicated to side-channel analysis.

In our experiments, we used a TPM implementation based on the design in [10]. The RTL description of the circuit was synthesized for the Virtex II Pro FPGA on the SASEBO board. For simplicity, our implementation consists of $K = 1$ hidden unit, having $N = 3$ inputs and a weight value range of $-L = -3 \ldots L = 3$. All weights are processed in parallel. We chose these parameters to ensure a low signal-to-noise-ratio for analyzing the weights in a single node and also to keep the total number $(2L + 1)^N = 7^3 = 343$ of possible weight combinations computationally feasible.

The data-dependent drawn power was measured with an digital oscilloscope (Agilent DSO6052) and a 1Ω-shunt placed in the supply line of the FPGA's inner core voltage. In this manner, we collected 600,000 individual power traces using uniform distributed random, but repeatable sequences of input data to the fixed-weight adaptation-inhibited TPM (as discussed in Section IV). After the measurement phase we partitioned the 600,000 traces into six groups of 100,000 traces each.

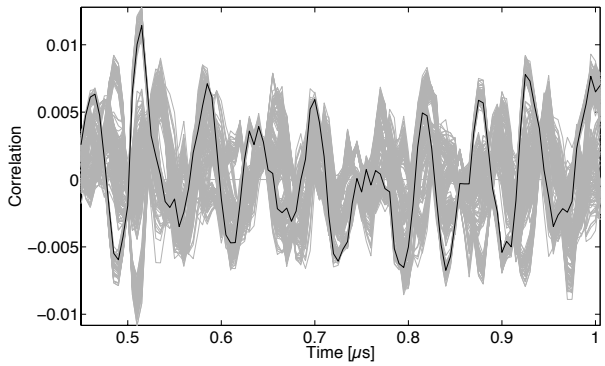

Fig. 3. Correlating measured and estimated power over time for all hypotheses (the correct hypothesis is plotted solid black)

A. Analysis using Hamming weight model

First, we analyze each trace group with the Hamming weight model *HW*. To this end, we compute the estimated power consumption by applying the Hamming weight model *HW* to the summation step of the calculation for our single hidden unit. We perform this for all weight vectors $\bar{w}$ (which are our hypotheses for the as-yet unknown secret vector) and the known current input $\bar{x}$.

For each input value, we have to test 343 hypotheses. Finally, we use the Pearson coefficient, which is a number between -1 and +1 that measures the degree of association between two statistical variables, to correlate the *measured* power for the input value $\bar{x}$ at time step t in the trace with all *estimated* power levels for the same input value (for all values of $\bar{w}$). The value $\bar{w}$ whose estimated power consistently has the highest correlation with the measured power, is most likely the correct value of the secret parameter vector.

Figure 3 shows the correlation results for the segment of time in which the summation of the parity calculation is processed. The grey bars indicate the maximum correlation achievable for all 343 choices of $\bar{w}$ for an input value $\bar{x}$. It turns out, however, that the values of $\bar{w}$ deemed to be most likely (having the highest correlation between measured and estimated power) are actually *not* the secret values of the actually used weights. For comparison, we have plotted the correlation of measured power with the estimated power of the *actual* secret values of $\bar{w}$ as a solid black line. For the six attacks we performed (each using a group of 100,000 traces), the correlation of the correct hypothesis is dominated in all cases by a false hypothesis (though not always the same one).

DPA thus cannot directly determine the correct key $\bar{w}$ when applying the Hamming weight model. However, it could aid an attacker by reducing the search space. We demonstrate this by a convergence analysis on our six group of traces (encompassing 100,000 power traces), Figure 4 shows exemplary

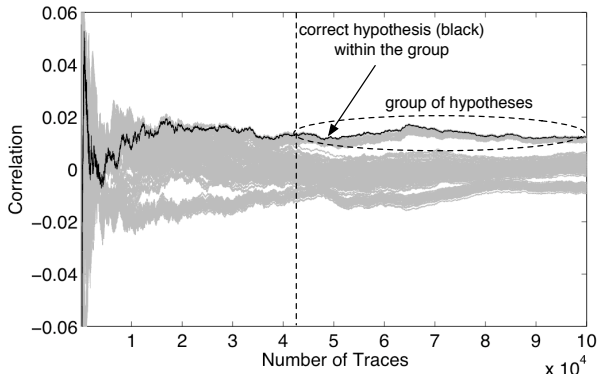

Fig. 4. Correlation of hypotheses for all values of $\bar{w}$ (grey) and of hypothesis for correct key (black) over the number of analyzed traces

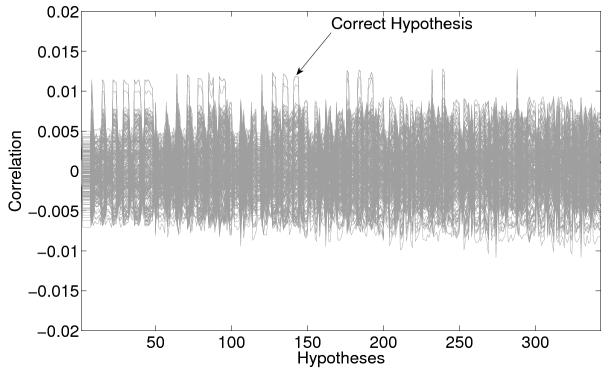

Fig. 5. Ranking of the different *HW*-based hypotheses with mark on the correct hypothesis

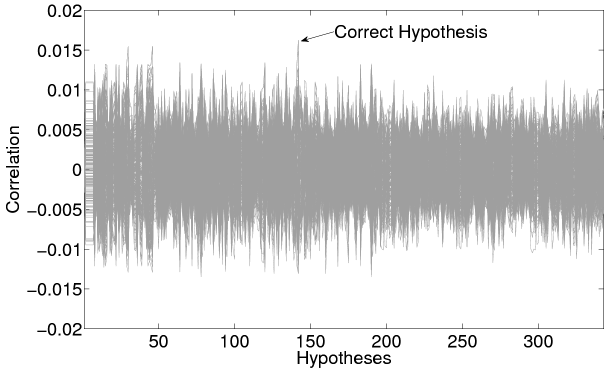

Fig. 6. Ranking of the different *HD*-based hypothesis with mark on the correct hypothesis

the result of one of the six power trace sets. The five other analysis results looks similar to the displayed. The drawback of such a convergence analysis is, that the computational effort is much higher than the effort for a normal differential power attack, why we tried the other approach first. This time, instead of just showing a *single maximum-correlation* hypothesis per time step, we depict the correlations of *all* 343 hypotheses over increasing numbers of used traces for the correlation analysis. In the first half of the experiment, with only a small number of traces considered, many hypotheses are deemed similarly likely. However, after roughly 42,000 traces, a group of hypotheses (marked by a dashed ellipse in Figure 4) clearly becomes more likely than the rest. Depending on the analyzed set of power traces this group contains 40...60 hypotheses, among them the *correct* values for $\bar{w}$ (this correlation is again plotted in black).

While Figure 4 is fine for getting an overview about the value of information contained in the data, but does not clearly show which ones of the 343 hypothesis are within the group, we will use a further representation. Figure 5 plots the correlation between the estimated power consumption and the real power consumption over the 343 different hypotheses. Individual hypotheses within the group of most-likely hypotheses have significantly higher correlation values than other hypotheses. The hypothesis with the correct weight values $\bar{w}$ is pointed out and is indeed among the hypotheses with the higher correlation values.

In summary, our first set of experiments shows that the Hamming weight model cannot accurately predict the power drawn by the actual hardware for given values of $\bar{w}$ and $\bar{x}$. But it can decrease the number of possible values for $\bar{w}$ (reduce the search space) at the cost of 100,000 measurements.

B. Analysis using Hamming distance model

In a second evaluation phase, we now use the Hamming distance model *HD* on each group of gathered traces to extract information leaking through the data-dependent power consumption. Building on the experience of the prior evaluation using the Hamming weight model, we proceed to directly analyze the correlation between the estimated power consumption, based on new HD-based 343 hypotheses, and the measured power consumption.

This time, for roughly half of our six analysis groups, the hypothesis with the highest correlation value actually *was* the correct hypothesis (shown in Figure 6). Even if it was incorrectly predicted (in the other half of our experiments), the correct hypothesis was among the group of the most-likely candidates (based on the correlation value). Furthermore, the group of most-likely candidates is now also much smaller (2 to 11 candidates in our experiments), reducing the search space further even if the correct hypothesis was not obvious right away. These results show that the more precise Hamming distance model is significantly better suited to a DPA attack on the TPM-based key exchange than the Hamming weight model.

VI. Conclusion and Future Work

In this paper, we have presented a strategy to attack a the tree parity machine algorithm using the differential power analysis and conducted experimental attacks on a FPGA implementation. To our knowledge, this is the first attempt at a side-channel analysis attack on a TPM-based cryptographic system. In particular, we were using Hamming weight and

Hamming distance models to get knowledge about the secret value of the weight vector $\bar{w}$, which is the only secret for the tree parity machine public key exchange. In our experiments, the latter model turned out to be more accurate in this scenario.

However, while we were partly successful with the Hamming distance model in determining the secret values of the weight vector in our experiments, our experiments also show that practical attacks on real TPM-based cryptographic systems will be very costly.

First, the accuracy of both of our Hamming models suffers from the absence of bijective non-linear functions (e.g., S-Boxes in AES and DES), which would have a significant Hamming distance between different intermediate values, and thus have a better exploitable data-dependent power consumption. Second, our attack targeted a very simple TPM cryptographic system. With more weights and nodes and a wider range of weight values, the number of hypotheses to test grows exponentially and will require a much larger computational effort than comparable attacks, e.g., on AES. Furthermore, an increased number of nodes will also produce more noise on the supply lines, especially when the nodes internally also use parallel compute architectures. This will make the capture of meaningful power traces even more difficult. Third, while our experiments captured hundred of thousands of power traces (since we artificially inhibited adaptation), a real system would synchronize after just a few hundred computations. Assuming a true random number generator for the initial weight values, there will be no similarities between different runs. Thus, it will not be possible to collect traces of the length required for a successful DPA attack, as every new synchronization run starts with different random hidden weight values.

Our results indicate that an increased number of nodes not only improves the resistance of TPM against collaborative attacks (as mentioned in [11]), but also hardens it against a standard DPA side-channel attack. Combined with the suitability of TPM for certain use-cases (such as the stream-cipher built on TPM's continuous key stream [11]), it thus becomes an even more attractive algorithm choice for resource-constrained environments.

Furthermore, the impact of the absence of bijective non-linear functions on the DPA resistance of a cryptographic system is a strong factor. Future research could study the inclusion of algorithms such as the TPM-based approach as an additional layer to conventional cryptographic systems to harden them against DPA attacks.

On the attack side, another angle could be to use the DPA just as a starting point. For example, [19] proposes an attack on the public key exchange based on a set of *cooperative* attackers, all starting at different states. Providing these attackers with more precise knowledge about the possible range of internal weight values could have the potential to speed-up the attack.

ACKNOWLEDGMENT

This work was supported by DFG grant HU 620/12, as part of the Priority Programm 1148 in cooperation with *CASED* (www.cased.de). All experimental measurements were done with the side channel experimental board SASEBO, which was developed by AIST and the Tohoku University, Japan, in the METI project.

REFERENCES

[1] T. Eisenbarth, S. Kumar, C. Paar, A. Poschmann, and L. Uhsadel, "A survey of lightweight-cryptography implementations," *IEEE Des. Test*, vol. 24, no. 6, pp. 522–533, 2007.

[2] S. Mangard, T. Popp, and M. E. Oswald, *Power Analysis Attacks - Revealing the Secrets of Smart Cards*. Springer, 2007.

[3] T.-H. Le, C. Canovas, and J. Clédière, "An overview of side channel analysis attacks," in *ASIACCS '08: Proceedings of the 2008 ACM symposium on Information, computer and communications security*. New York, USA: ACM, 2008, pp. 33–43.

[4] T. S. Messerges, E. A. Dabbish, and R. H. Sloan, "Investigations of power analysis attacks on smartcards," in *USENIX Workshop on Smartcard Technology*, 1999, pp. 151–162.

[5] M. Aigner and E. Oswald, "Power analysis tutorial," Institute for Applied Information Processing and Communication, University of Technology Graz - Seminar, Tech. Rep., 2000.

[6] I. Kanter, W. Kinzel, and E. Kanter, "Secure exchange of information by synchronization of neural networks," *Europhysics Letters*, vol. 57, no. 1, pp. 141–147, 2002.

[7] M. Volkmer and S. Wallner, "Tree parity machine rekeying architectures," *IEEE Trans. Comput.*, vol. 54, no. 4, pp. 421–427, 2005.

[8] A. Klimov, A. Mityagin, and A. Shamir, "Analysis of neural cryptography," in *ASIACRYPT '02: Proceedings of the 8th International Conference on the Theory and Application of Cryptology and Information Security*. London, UK: Springer-Verlag, 2002, pp. 288–298.

[9] A. M. Allam and H. M. Abbas, "Improved security of neural cryptography using don't-trust-my-partner and error prediction," in *IJCNN'09: Proceedings of the 2009 international joint conference on Neural Networks*. Piscataway, NJ, USA: IEEE Press, 2009, pp. 1900–1906.

[10] S. Mühlbach and S. Wallner, "Secure and authenticated communication in chip-level microcomputer bus systems with tree parity machines," in *International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation*, 2007, pp. 201–208.

[11] S. Mühlbach and S. Wallner, "Secure communication in microcomputer bus systems for embedded devices," *J. Syst. Archit.*, vol. 54, no. 11, pp. 1065–1076, 2008.

[12] T. Chen and S. H. Huang, "Tree parity machine-based one-time password authentication schemes," in *International Joint Conference on Neural Networks*, 2008, pp. 257–261.

[13] D. Hu and Y. Wang, "Security research on wimax with neural cryptography," in *ISA '08: Proceedings of the 2008 International Conference on Information Security and Assurance (isa 2008)*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 370–373.

[14] B. Saballus, S. Wallner, and M. Volkmer, "Secure group communication in ad-hoc networks using tree parity machines," in *KiVS 2007*, T. Braun, G. Carle, and B. Stiller, Eds. Bern, Switzerland: VDE Verlag, 2 2007, pp. 457–468.

[15] S. Wallner, "Designing low-cost cryptographic hardware for wired- or wireless point-to-point connections," in *Advances in Information Security and Its Application Third International Conference, ISA 2009, Seoul, Korea, June 25-27, 2009*. Springer, 2009, pp. 1–10.

[16] M. Volkmer, "Entity authentication and authenticated key exchange with tree parity machines," Cryptology ePrint Archive, Report 2006/112, 2006, http://eprint.iacr.org/.

[17] P. C. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *CRYPTO '99: Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*. Springer, 1999, pp. 388–397.

[18] [Online]. Available: www.rcis.aist.go.jp/special/SASEBO/index-en.html

[19] L. N. Shacham, E. Klein, R. Mislovaty, I. Kanter, and W. Kinzel, "Cooperating attackers in neural cryptography," *Phys. Rev. E*, vol. 69, no. 6, p. 066137, Jun 2004.