Towards Electrical, Integrated Implementations of SIMPL Systems

Ulrich Rührmair Qingqing Chen Paolo Lugli Ulf Schlichtmann
Martin Stutzmann György Csaba

June 9, 2009

1 Abstract

This paper discusses the practical implementation of a novel security tool termed *SIMPL system*, which was introduced in [1]. SIMPL systems can be regarded as a public key version of physical unclonable functions (PUFs). Like the latter, a SIMPL system S is physically unique and non-reproducible, and implements an individual function F_S. In opposition to a PUF, however, a SIMPL system S possesses a publicly known numerical description, which allows its digital simulation and prediction. At the same time, any such simulation must work at a detectably lower speed than the real-time behavior of S. As argued in [1], SIMPL systems have certain practicality and security advantages in comparison to PUFs, certificates of authenticity, physically obfuscated keys, and also to standard mathematical cryptotechniques.

In [1], definitions, protocols, and optical implementations of SIMPL systems were presented. This manuscript focuses on concrete electrical, integrated realizations of SIMPL systems, and proposes two potential candidates: SIMPL systems derived from special SRAM-architectures (so-called "skew designs" of SRAM cells), and implementations based on Cellular Non-Linear Networks (CNNs).

2 Introduction

Physical Unclonable Functions (PUFs) are a relatively young, emerging cryptographic primitive [2, 3, 4, 5, 6, 7]. However, one potential downside of PUF-based protocols is that they usually require a previously shared piece of information (typically some challenge-response-pairs) that was established in a joint set-up phase between the communicants. Alternatively, an online connection to a trusted authority at the time of the protocol execution must be employed. In this particular structural aspect, PUFs are resemblant of classical private key systems.

In this paper, we are concerned with an alternative security tool called *SIMPL systems*, which is a public key version of standard PUFs. SIMPL systems have been introduced in [1]. The acronym SIMPL stands for "SIMulation Possible, but Laborious", and hints at the critical security feature of these structures. A physical system S is called a *SIMPL system* if the following holds:

1. It is possible for everyone to numerically simulate and, thus, to predict the physical behaviour of S with very high accuracy. The basis of the simulation is an individual description $D(S)$ of S, and a generic simulation algorithm Sim, which are both publicly known.

2. Any sufficiently accurate numerical simulation — as well as any arbitrary physical emulation of S — is slower than the real-time behavior of S. Determining the system's behavior by an actual measurement on the original system S works detectably quicker than any other approach.

3. It is difficult to physically reproduce or clone S.

Put together in one sentence, the holder of a SIMPL system S can compute a publicly known, publicly computable individual function F_S faster than anyone else. Applying the familiar public key terminology to this situation, one could state that the numeric description $D(S)$ essentially serves as a public key, while the physical system S constitutes an equivalent to a private key. This "private key", however, is a physically irreproducible structure, which contains no secret information at all. This leads to several significant security advantages, which have been discussed in [1].

One critical question is certainly how SIMPL systems can be implemented in practice. We suggest two variants based on integrated electrical circuits in this publication: Firstly, special SRAM-memories based on a newly developed "skew" design, which leads to fuzzy memory cell behavior at quickly varied operational voltages. Secondly, we propose analog circuits known as Cellular Non-Linear Networks (CNN), whose cells evolve over time in an analog, highly parallel fashion. This can help them to outperform classical architectures on certain computational tasks, as is required from SIMPL systems.

The rest of this manuscript is organized as follows: In section 3 we cite and discuss the formal definition of SIMPL systems of [1]. Section 4 provides one example protocol and briefly discusses applications and advantages of SIMPL systems. In section 5 we treat the implementation of SIMPL systems by Cellular Non-Linear Networks. Section 6 introduces SIMPL systems based on special SRAM architectures. We conclude the paper in section 7.

3 Definition of SIMPL Systems

The following definition of SIMPL systems has been provided in [1].

Definition 3.1 ((t_C, t_{Ph}, ϵ)-SIMPL Systems). *Let S be a physical system mapping challenges C_i to responses R_i, with $\mathbf{C}$ denoting the finite set of all possible challenges. Let furthermore t_{max} be the maximum time (over all challenges $C_i \in \mathbf{C}$) which it takes until the system has generated the corresponding response R_i. S is called a (t_C, t_{Ph}, ϵ)-SIMPL SYSTEM if there is a numerical string $D(S)$, called the description of S, and a generic computer algorithm* Sim *such that the following conditions are met:*

1. *For all challenges $C_i \in \mathbf{C}$, the algorithm* Sim *on input*

$$\big(C_i,\, D(S)\big)$$

outputs R_i in feasible time.

2. *Any cryptographic adversary Eve, who is bound to practically feasible probabilistic Turing computations and practically feasible physical actions, will succeed in the following* **security experiment** *with a probability of at most ϵ:*

   (a) *Eve is given the numerical description $D(S)$ and the code of the algorithm* Sim *for a time period of length t_C.*

(b) Within the above time period t_C, Eve is given physical access to the system S at adaptively chosen time points, and for adaptively chosen time periods. The only restriction is that her access times must add up to a total of at most t_{Ph}. After the access times have ended, she does not have physical access to S anymore.

(c) After period t_C has expired, Eve is presented with a challenge C_{i_0} that was chosen uniformly at random from the set $\mathbf{C}$, and must output a value V_{Eve}.

We say that Eve succeeded in the above experiment if the following conditions are met:

(i) $V_{Eve} = R_{i_0}$.

(ii) The time that Eve needed to output V_{Eve} after she was presented with C_{i_0} is at most $2 \cdot t_{max}$.

Please note that the said probability of ϵ is taken over the uniformly random choice of $C_{i_0} \in \mathbf{C}$, and the random choices or actions that Eve might take in steps 2a, 2b and 2c.

Some remarks on the definition are in order.

Security Model. Let us start by briefly discussing the security model of the definition. In practice, an adversary Eve can gather information about S in essentially two ways. Firstly *computationally*, by analyzing challenge-response-pairs (C_i, R_i) and by analyzing the algorithm Sim and the description $D(S)$. The CRPs may either stem from eavesdropping on protocols, or they may be computed by the adversary hinself via the algorithm Sim and the description $D(S)$. These possibilities are reflected in item *2a* of the definition. Secondly, Eve may *physically* measure arbitrary features of the system S at some point. For example, she might try to obtain some physical characteristics or internal parameters of the system which are not easily deducable from knowing many CRPs, but which could speed up her simulation. This possibility is covered in item *2b*. The model tries to reflect real-world situations, for example if S was used in mobile systems for identification purposes.

Immunity against Full Read-Out. It follows from Definition 3.1 that for any SIMPL system S, it must be impossible to measure the values R_i for *all* possible parameters $C_i \in \mathbf{C}$ within the timeframe t_{Ph}. Otherwise, Eve could create an exhaustive lookup-table for all possible values R_i during step 2b, which would enable her to succeed in the described experiment. Hence, for any SIMPL system either the set of possible measurement parameters $\mathbf{C}$ must be very large (for example exponential in some system parameter) and/or successive read-outs can only be carried out relatively slowly.

Immunity Against Cloning. Please note further that Definition 3.1 implies that previous physical access and a number of known Challenge-Response-Pairs of S must not enable Eve to do one of the following:

1. Build an *exact physical clone* S' of the system S, for which

$$R_i = R'_i \quad \text{for (almost) all } C_i \in \mathbf{C},$$

and for which the evaluation of the R'_i works comparably quickly as by an experiment on S.

2. Build a *functional physical clone* S' of S, which may be a physical system of a possibly very different structure or different lengthscales than S, that enables Eve to determine the values R_i for (almost) all $C_i \in \mathbf{C}$ correctly and comparably quickly as by experiment on S.

3. Build a *digital clone*, which is a computer algorithm Alg that numerically computes the values

$$Alg(C_i) = R_i$$

for (almost) all $C_i \in \mathbf{C}$ comparably quickly as by an experiment on S.

The inability for *digital cloning* implies a number of non-trivial requirements: Firstly, it logically includes the immunity against full read-out that we discussed earlier. Secondly, it implies that the behaviour of S cannot be learned by a machine learning algorithm that has a very rapid prediction phase, which works on a comparable timescales as the real-time behavior of S. Thirdly, and most generally, it implies that the simulation of S on the basis of $D(S)$ cannot be split into a possibly laborious precomputation phase independent of a concrete challenge, and a specific computation phase that very rapidly determines R_i once C_i is given.

In the sequel, we will sometimes refer to the immunity of S against cloning also as the unreproducibility or the uniqueness of S.

Different Adversarial Scenarios. The definition leaves to some extent open which specific resources Eve may employ during her attack. There are several meaningful scenarios, leading to different security notions.

1. Consumer Security: Eve is assumed to be a private person, possibly very educated in cryptographic and security matters, but with a budget not exceeding one million dollars.

2. Organization Security: Eve is an organization with very strong financial resources, possibly up to billions of Dollars, but with some significant constraints concerning impending legal prosecution.

3. Technological Security: We assume that Eve is allowed to use basically unlimited fincancial resources, and faces no restrictions other than those induced by current technology.

When we say that a SIMPL system is secure in one of the above scenarios, we mean that it remains secure in the sense of Definition 3.1 if Eve is allowed the described resources. Which type of security we seek strongly depends on the intended application. A SIMPL system that is not technologically secure, but consumer secure might still find very fruitful applications in the consumer market. One should have this fact in mind, and not aim for technological security only when designing SIMPL systems.

For a full discussion of the definition and the underlying adversarial model, see [1]. In particular, we argue in this reference why the usual distinction between polynomial and exponential time should not be applied in the definition of SIMPL systems, which only allows us to employ seemingly handwaving notions like "practically feasible" or "infeasible". In particular, there is no formal "computational" model at hand which could help us to formalize Eve's *physical* actions during her physical access periods t_{Ph}.

4 Protocols and Applications

We will now quote one exemplary protocol that can be realized by SIMPL systems in order to illustrate their working principle [1]. A few applications and the advantages of SIMPL systems are briefly discussed, too.

4.1 Identification by SIMPL Systems

We assume that Alice, who holds an individual SIMPL system S, has put $D(S)$, Sim, t_{max} and a description of $\mathbf{C}$ in a public register (we will not discuss PKI-related problems such as [10] here). Now, she can prove her identity to an arbitrary second party Bob as follows [1]:

Protocol 4.1: IDENTIFICATION OF ENTITIES BY SIMPL SYSTEMS

1. Bob obtains the information $D(S)$, Sim, t_{max}, and $\mathbf{C}$ associated with Alice from the public register.

2. Bob sends a number of randomly chosen challenges $C_1, \ldots, C_k \in \mathbf{C}$ to Alice.

3. Alice determines the corresponding responses $R_1, \ldots, R_k$ by experiment on her SIMPL system S, and returns them immediately to Bob.

4. Bob receives values $V_1, \ldots, V_k$, and measures Alice's response time (i.e. the time between the two events of sending $C_1, \ldots, C_k$ and receiving $V_1, \ldots, V_k$). If this time is above the threshold $2 \cdot t_{max}$, he aborts the protocol.

5. Bob checks through simulation by the algorithm Sim if for all $i = 1, \ldots, k$,

$$V_i = R_i.$$

   If this is the case, Bob believes Alice's identity, otherwise not.

Security. As usual, k is the security parameter of the protocol. In a nutshell, the protocol works because Eve is unable to determine the values R_i for randomly chosen C_i comparably quickly as Alice, provided that: (i) The lifetime of the system S (and the period since $D(S)$ was made public) does not exceed t_C, and (ii) Eve's accumulated physical access times to S do not exceed t_{Ph}. In that case, Eve's probability to succeed in the protocol without possessing S are less or equal to ϵ^k.

Practicality. Bob can improve his computational efficiency by verifying the correctness of the responses R_i merely for a randomly chosen, smaller subset of $\{1, \ldots, k\}$. If necessary, possible network and transmission delays can be compensated for in advance by amplifying the absolute time gap between Eve and S through feedback loops (see discussion in section 3). Also the asymmetry between checking a solution and computing a solution may be exploited in future protocols (see section 6.3 of [1]).

4.2 Applications and Advantages of SIMPL Systems

Straightforward applications of the above identification protcol include the following [1]:

(i) Identification of hardware and computer systems.

(ii) Secure labeling of valuable items, such as branded products, pharmaceuticals, passports, bank notes, credit cards, and the like.

(iii) Unclonable (copy protected) representations of digital content and software, digital rights management.

(iv) Tamper sensitive hardware environments.

The upside of using SIMPL systems in these situations over standard mathematical cryptotechniques or alternative approaches such as certificates of authenticity [12] or PUFs has been discussed in detail in [1]. It includes: (i) SIMPL systems do neither contain nor constitute any sort of secret binary information. This makes them naturally immune against any side channel, invasive or malware attack. (ii) They allow protocols that are independent of the standard, unproven number theoretic assumptions (factoring, discrete log). (iii) They have strong practicality advantages over COAs and PUFs, since structurally they are a public key version of PUFs. They allow new DRM in connection with TPMs, or labels that can be read out digitally over long distances, and can be verified offline at the same time [1].

These assets make them a worthwhile target for future investigations. In particular, it would be important to find electrical, integrated implementations, an issue which was left open in [1]. Over the next sections, we will discuss two promising candidates.

5 Cellular Non-Linear Networks

5.1 Introduction and General Idea

A first *electrical and on-chip* candidate for SIMPL systems are Cellular Non-Linear Networks (CNNs) [22]. If successfully implemented, they would result in a *technologically secure* SIMPL system (see page 4).

CNNs are analog computing arrays with a regular, periodic, cellular structure. The cells are characterized by a dynamical state variable, and their time evolution depends on their own internal state and on the inputs from their neighbouring cells. On an abstract level, their behavior is given and determined by so-called templates, which in the simpliest case are real-valued matrices. On a circuit level, it is given by the transistor architecture of a cell, which implements the behavior specified by the templates.

More specifically, each cell is characterized by a dynamical state variable x, which obeys the following, ordinary differential equation (ODE):

$$\dot{x}_{ij} = -x_{ij} + \sum_{k,l} \mathbf{A}_{i,j,k,l} y_{kl} + \sum_{k,l} \mathbf{B}_{i,j,k,l} u_{kl} + z_{ij}$$

i.e. the time derivative of the state variable (for the cell with i, j indices) depends on the y output of the neighboring cells (denoted by the k, l indices) via a the $\mathbf{A}$ cloning templates. Each cell has a bias (z) and inputs, which are coupled by the $\mathbf{B}$ template to the equation.

As a mathematical model, CNNs are very general; for example, cellular automata [14] can be interpreted as a special CNN which operates on discrete variables in discrete time (and where rules replace the ODE-based description). CNNs are also known to be Turing-complete [15]. CNNs often have multiple layers, and these layers are also coupled to each other via **B** templates.

Due to their analog and highly parallel architecture, CNNs have a remarkable computing power and efficiency. Already in 2004, a state-of-the-art programmable, commercially available CNN in a 0.35-μm standard CMOS technology exhibited peak computing figures of 330 GOPS [23] (or 3.6 GOPS/mm^2 and 82.5 GOPS/W in terms of area and power consumption). These numbers are yet excelled by non-programmable CNNs, which we propose for use as SIMPL systems. In specialized tasks, it is known that CNNs can outperform digital computers by a factor of up to 1,000 [24, 25]. CNNs are the largest analog circuits, with the CNN referred to above [23] containing 3.75 million transistors.

A further important property of CNNs is that their functionality is especially sensitive to the inevitable variations in the fabrication process, unless special countermeasures are taken. This can make the function F_S computed by a CNN S truly unique. At the same time, since CNNs are integrated electrical systems, dedicated on-chip measurement circuitry can determine the fabrication mismatches, and deliver a sufficiently detailed description $D(S)$ to simulate F_S. Such types of self-measuring cells are already today in standard use for calibration purposes [26]. Furthermore, it is known that there is a stable regime where the fabrication mismatches determine the CNN behavior, and where they override circuit noise and temperature variations [27, 28]. Altogether, said properties makes CNNs quite interesting candidates for SIMPL systems.

5.2 Implementation

We propose two concrete candidates for CNN-based SIMPL systems. Firstly, CNNs employed for image processing tasks [25, 30], which are known to outperform classical architectures by a factor of 10 – 100. Another attractive option is a template and circuit-design that has been recently devised in our group [29]. It is inspired by the high internal complexity of optical PUFs [2], in whose time evolution many internal scattering components interact in parallel, leading to a high computational complexity and to laborious simulatability. Our template has the remarkable property that it effectively transfers optical behavior onto a CNN (i.e. onto an electrical integrated circuit), which then behaves quasi-optical, that is, similar to an optical system. In particular, the electrical current flowing through a certain reference point in each CNN-cell is equivalent to the local light intensity in an optical interference reference system.

The upcoming figures provide the templates and cell architecture of this 3-layer CNN, as well as simulation results that confirm the quasi-optical behavior. Figure 1 shows the templates and the interaction structure of the proposed 3-layer CNN. Figure 2 illustrates the circuit-level design. Figure 3 provides simulation data which shows the quasi-optical interference patterns in the linear (left) and non-linear/mismatched case (right). Figure 4 illustrates that local changes in the structure propagate globally. This further illustrates the quasi-optical nature and the high computational complexity of the structure: Its evolution involves many interacting subunits in parallel.

The described CNN-design seems particularly suited as SIMPL system because its quasi-optical behavior fosters pairwise interaction between the cells throughout the structure. This leads to a particularly strong, inherent parallelism, which will be costly to simulate on digital architectures. Furthermore, as we could show in simulations, the behavior of the quasi-optical SIMPL automatically shifts into a non-linear, highly complex regime through the occurring manufacturing mismatches, which can be exploited even better for our purposes. In opposition to three-dimensional optical PUFs, its description $D(S)$ can be determined by in-built on chip measurement circuitry.

Another very important characteristics of our circuit that its behavior is sensitive, but is not chaotic. Chaotic circuits are well known [16] and several CNN templates are known to realize

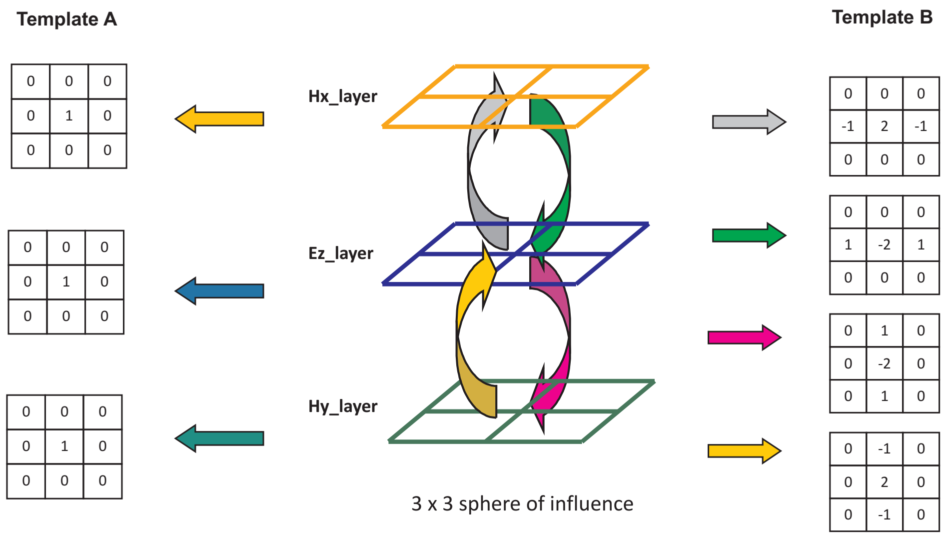


Figure 1: Templates and interaction structure of our 3-layer CNN-SIMPL system.

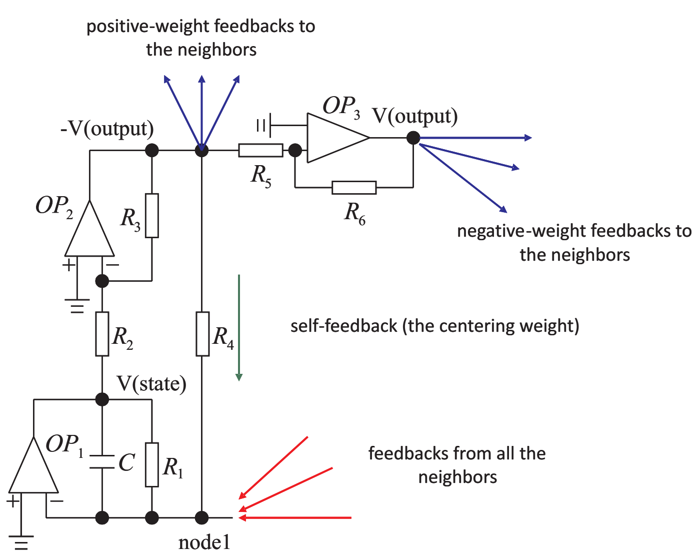
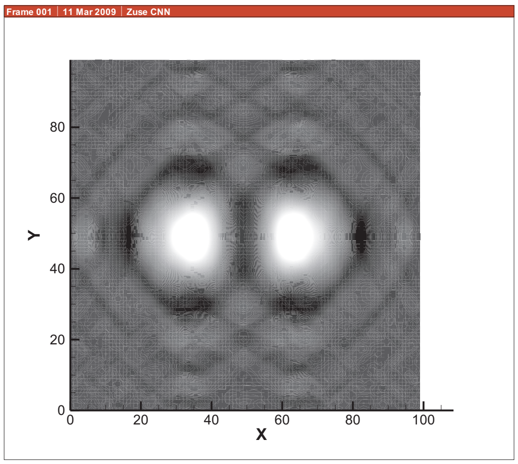


Figure 2: Circuit level design of our proposed CNN-SIMPL system.



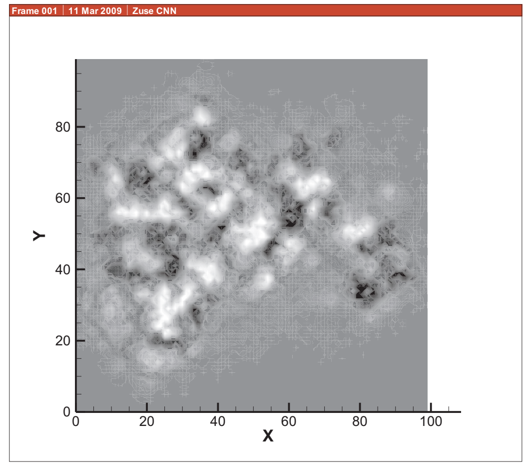


Figure 3: Simulated behavior of the CNN-SIMPL system. The brightness levels illustrate the currents at a fixed reference point in each cell within a 100 × 100 cell structure. Left: Linear case, without fabrication mismatches, and with two excitation sources. Right: Non-linear case, resulting from fabrication mismatches, again two excitation sources. Both pictures nicely show the quasi-optical interference behavior. The non-linear case obviously provides a much more complex and much richer regime, which is suited best for our purposes.

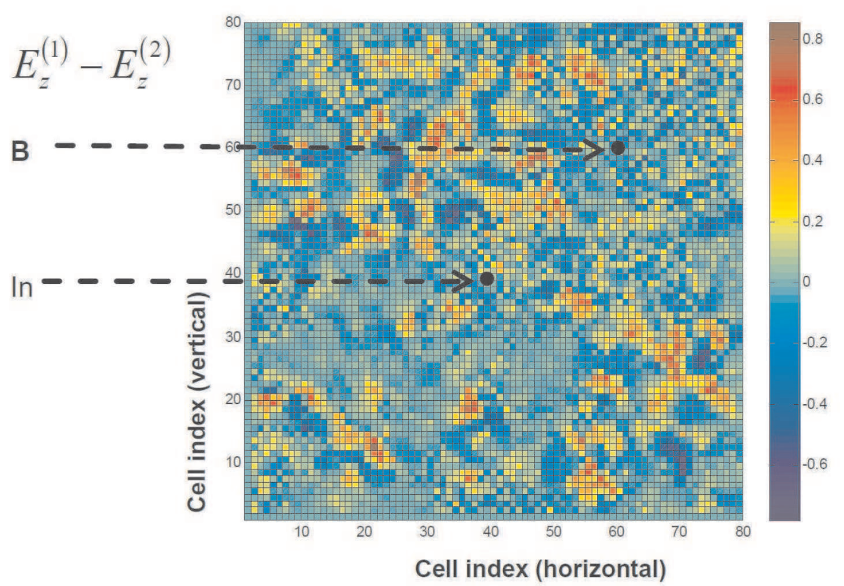


Figure 4: Globale sensitivity of the proposed CNN design to a local change of the template. Changing only a single template at a particular position (denoted by B in the figure), even far away from the input exciting the structure (marked as In), will alter the global behavior of the circuit detectably. The figure shows a difference map of the values E_z^1 and E_z^2 obtained by two simulations, one for the original value of the templates, the other for one template value in position B altered.

chaos [17, 18, 19]. The time trajectories of a chaotic system are irreproducible in a real physical environment and are hence unsuited as a SIMPL system.

Security Aspects. A 100×100 cell CNN with our architecture leads to the following specific numbers: It requires a description $D(S)$ containing about $10^4 \cdot 19$ template values, which is about 100 kB of information. In order to simulate the real-time evolution which the CNN undergoes in a few microsecond time frame, 10^4 coupled differential equations need to be solved (i.e., one for each cell).

Non-programmable CNNs require about 10 times less transistors per cell than programmable CNNs, and they operate faster due to the simple, resistive interconnections of their cells. By increasing the size of our non-programmable CNN, we can easily shift into regions where commercially programmable CNNs are no longer available.

6 SIMPL Systems from Special SRAM Memories

6.1 Introduction and General Idea

One practical and stable, but at most consumer secure SIMPL candidate will be presented in this section. It is based on a special design of SRAM memories, which we call *"skew design"*. Its basic idea is to design the SRAM-cells such that they exhibit varying behavior in different operational voltage regions. Some of the cells (cells of type 1) will function properly over the whole operational voltage range. Others, of type 2, will possess stable read operations, but exhibit (intended) write failures whenever the operational voltage VDD is above a certain threshold. This means that in these VDD regions, the content of the cell is not changed or affected by write procedures. Below the threshold, however, the write operation in cells of type 2 functions properly. Finally, there are cells of type 3, which contain a fixed bit value (0 or 1). It has been hardwired into them already in their fabrication, and their content cannot be changed by any write operation at all, regardless of the applied operational voltage.

Now, imagine an SRAM-memory M where cells of the described three types are randomly distributed or mixed. We call such a memory a "skew memory". Imagine further that on the basis of M, we build a larger hardware system S, which repeats the following feedback loop l times at maximal operational speed.

Feedback loop, iteration i:

1. Write bitvalues $b_1^i, \dots, b_k^i$ into the addresses $WR_1^i, \dots, WR_k^i$ of M
2. Read out the bit values $B_1^i, \dots, B_m^i$ from the addresses $READ_1^i, \dots, READ_m^i$
3. Switch to operational voltage $VDD(i)$
4. Determine the parameters necessary for the next iteration, namely $b_1^{i+1}, \dots, b_k^{i+1}$, $WR_1^{i+1}, \dots,$ WR_k^{i+1}, $READ_1^{i+1}, \dots, READ_m^{i+1}$, $VDD(i+1)$, as a pseudo-random function of the values $B_1^i, \dots, B_m^i$ obtained in step 2.

S is depicted schematically in Fig. 5. In order to associate a global input and a global output with S, we may say that the values $b_1^0, \dots, b_k^0$, WR_1^0, $\dots$, WR_k^0, $READ_1^0, \dots, READ_m^0$, $VDD(0)$ that are necessary to start the loop, constitute its global input. After the last of the l iterations, the values $B_1^l, \dots, B_m^l$ can serve as the global output of S. Alternatively, one may define the global output to be a function (e.g. a hash function) of the values $B_1^{l-q+1}, \dots, B_m^{l-q+1}$, $B_1^{l-q+2}, \dots,$ $B_m^{l-q+2}, \dots, B_1^l, \dots, B_m^l$ that occured in the last q iterations of the loop. In this sense, we can interpret the behavior of S as a function F_S mapping global inputs to outputs.

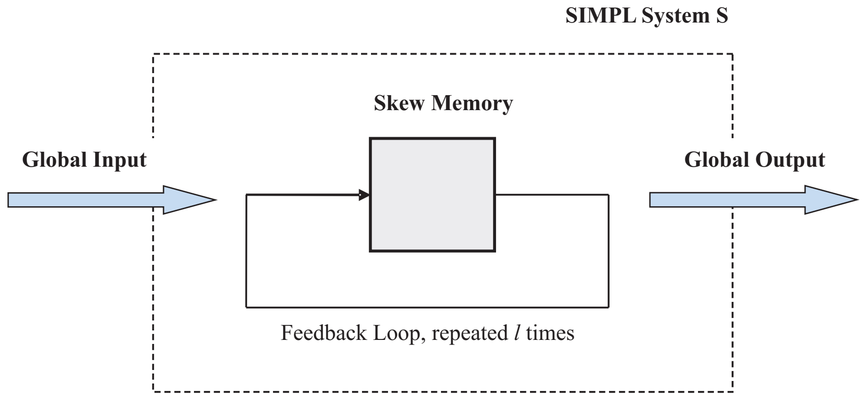


Figure 5: Schematic illustration of the input–output behavior of S and of the function F_S.

Then, F_S has the following properties:

(i) F_S can be individualized by changing the design of the memory M. To that aim, for example memory cells of type 3 (fixed bitvalues) can be distributed randomly over the memory in a final fabrication step.

(ii) If the distribution of the cells of type 1, 2 and 3 is known, the function F_S can be simulated digitally.

(iii) The simulation of F_S on a standard architecture will be slower than the real-time computation of F_S by S. Also configurable hardware or ASICs that are not based on a skew design will have a speed disadvantage. In both cases, the speed gap will only be a constant factor, however.

(iv) If the special skew design of SRAM cells is legally protected, then an adversary needs his own chip foundry to produce a hardware system that implements F_S comparably quickly, since ordering ASICs with a skew design will be legally prohibited.

The above properties qualify S as a consumer secure SIMPL system. We will discuss the practical implementation over the next section.

6.2 Implementation

A concrete skew design developed in our group [13] is illustrated in Fig. 1a), with width and length specified beside each transistor. The functionality of the design based on TSMC 0.18 μm technology has been succssfully verified with Spectre [31] simulations. The corresponding results are illustrated in Figure 7. In our case, $VDD_{min} = 1.4\,V$, $VDD_{max} = 1.7\,V$, and $VDD_{funcmin} = 1.58\,V$.

The memories, which will all share the same layout, can be individualized towards the end of manufacturing by fixing the content of some individually chosen cells to certain values. This means that the resulting structure will not be *manufacturer resistant* in the sense of [4], but will at least require a fraudster to possess its own chip foundry. The common SRAM-cell arragement will be contained in the general simulation algorithm Sim, and the individual description $D(S)$ consists of the cells that have been fixed to certain values. Please note that the described individualization

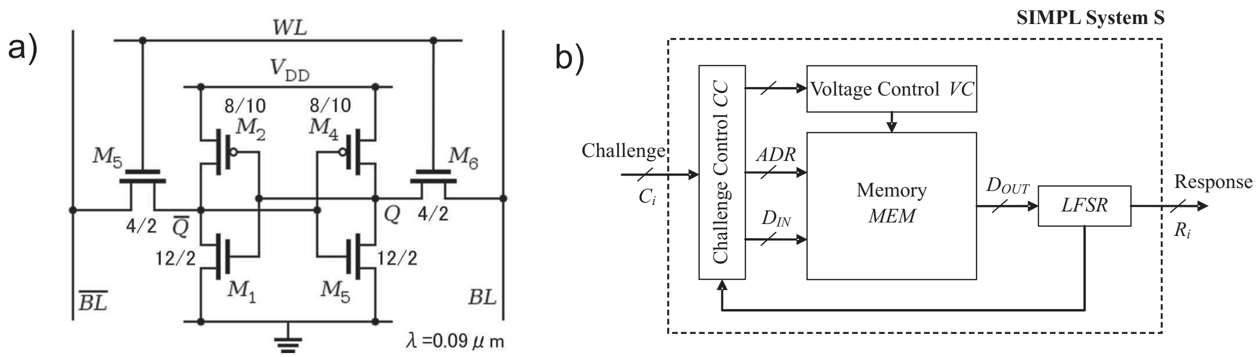


Figure 6: (a) The SRAM cell layout. (b) The basic operation cycle of the SRAM-SIMPL system.

can be carried out on the basis of a pseudorandom number sequence, which means that a *short, few-hundred bit long random seed s* suffices as $D(S)$.

The basic implementation of the feedback-loop is sketched in Fig. 1b). The implementation of the pseudo-random generator is carried out by an LSFR, since that PRNG works very quickly. Computationally laborious PRNGs might be implemented more quickly by a fraudster, who would thereby regain some of his speed disadvantage. Please note that we do not require a PRNG with cryptographic security in this application, but merely a PRNG with a long periodicity, such that as many memory cells as possible are at least once written to or read from in the feedback loop.

The *relative* speed advantage of the real system can be further amplified by activating and writing into multiple word lines during one write cycle. Due to the skew design, the same value written in several lines will not necessarily result in the same cell content. Based on the simulation data we obtained, we estimate that the relative speed advantage of a SIMPL SRAM memory will be a factor on the order of 10, even compared to dedicated, configurable hardware such as FPGAs. At the same time, since all operations on the SRAM-memory are fully digital and well-defined, the content of the memory can be precisely simulated and predicted.

Compared to optical SIMPLs [1] and CNN-SIMPLs, the great advantage of the SRAM-variant is its practicality and stability. It can be implemented relatively cheaply, integrated in existing systems, and requires only very short descriptions $D(S)$. This comes at the cost of losing their technological security, and exchanging it against consumer security (see page 4). Nevertheless, this seems acceptable in many applications. Please note that for integrated PUFs and SIMPLs, *some* level of manufacturer trust will always be required: It is difficult to prevent that a fraudulent manufacturer implements a PRNG with a seed known to him and distributes it to innocent customers, instead of producing a real, manufacturer resistant PUF.

Security Aspects. Let us *very* briefly discuss a few security relevant aspects. A fraudster who wants to imitate the skew SIMPL systems without a skew architecture has a number of basic possibilities.

First of all, he may try to implement the feedback loop in full logic, that is, without any memory cells at all. His hope may be that pure logic operations work faster than memory read and write steps, and that he can so outperform (or at least match) the speed of the original SIMPL. However, if the memory is sufficiently large, then the construction of such a pure logic will be prohibited by size and complexity constraints.

This means that the faker needs to employ some sort of memory in his attempts. SRAM

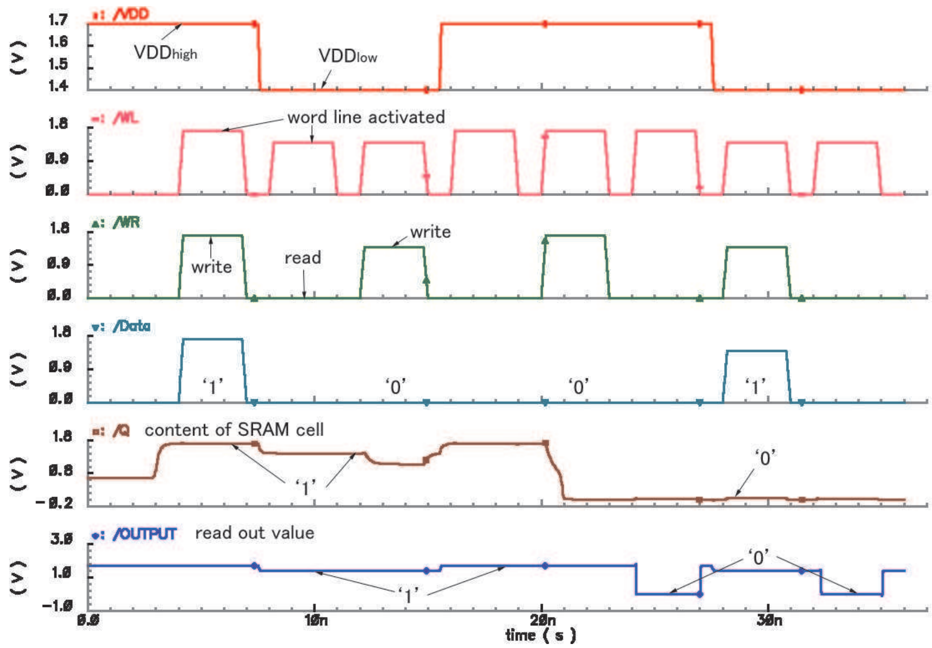


Figure 7: Spectre Simulations confirm the desired behavior: Write failures occur at certain voltages, meaning that the content of the SRAM cell remains unchanged in the WRITE operation. At the same time, the READ operation functions properly at all voltages.

memories are, in general, the fastest currently available technology, meaning that the faker should use SRAM cells, too. If he cannot rely on skew cells, one straightforward approach may be a system S_{Fake} as described in Figure 8. There, three standard memories M^*, M_1 and M_2 and one logic block are used in order to simulate one skew memory M_{Skew}. All of them have the same size (i.e. number of cells) as M_{Skew}.

The purpose of M^* is to store the same content as M_{Skew} during all of its operation time. The purpose of the memories M_1 and M_2 is to store the distribution of cells of type 1, 2 or 3 in M_{Skew}. For describing the type of a cell in address ADD of M_{Skew}, we need two bits (for the three different types). These two bits are distributed in the same addresses of the memories M_1 and M_2: M_1 stores in ADD the first bit of the description, whereas M_2 stores in ADD the second bit.

Under these provisions, S_{Fake} can simulate one write procedure of bit b under operational voltage VDD into address ADD of M_{Skew} as follows: The two bits describing the cell type of M_{Skew} in address ADD are read from the address ADD in M_1 and M_2. They are fed into the logic, together with the operational voltage VDD. The logic determines whether a write operation under the described circumstandes (type of cell and given VDD) would be successful. If yes, then the bit b will be written into ADD in the memory M^*. If not, nothing happens, and no write operation will be executed.

The described construction ensures that M^* always carries the same content as M_{Skew} would carry under a similar operation. At the same time, however, it requires more Read/Write operations plus the additional logic steps, leading to a constant speed-up factor. If also multiple word lines are activated in M_{Skew} in each write step, this constant speed-up factor will be above a value of 2, meeting definition 3.1.

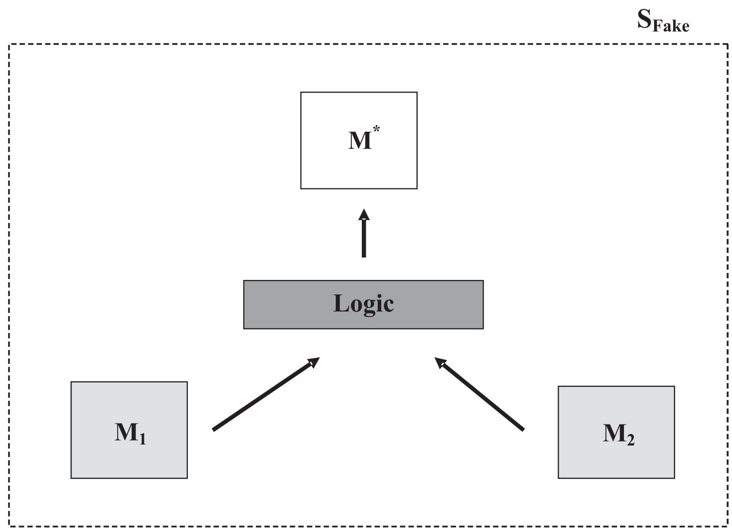


Figure 8: A system S_{Fake} designed to imitate a skew memory M_{Skew}.

7 Conclusions

SIMPL Systems are a novel security concept, which can be regarded as a public key version of physical unclonable functions [1]. Structurally, they function like a private/public key cryptosystem, with the notable difference that the equivalent to the private key is a physically hard-to-reproduce structure, which does not contain any secret information at all. This leads to critical security and practicality advances. In this paper, we reviewed the basic concepts presented in [1], but mainly focused on promising IC-based implementations of SIMPL systems.

Our first idea was to employ large, analog computing arrays as SIMPL systems. They evolve in parallel and by exchanging analog signals between their subunits, creating a significant computational power and complexity. At the same time, the arrays can be designed to strongly depend on fabrication mismatches, making the function which they implement individual and unique. As a concrete implementation, we suggested to use cellular, non-linear networks with special templates, since they are the largest currently known analog circuits with up to millions of transistors. We proposed one concrete design, and evaluated its working principle in several simulations. If implemented successfully, CNN-based SIMPL systems may eventually lead to technologically secure SIMPL systems.

Our second idea was to use special ASICs as SIMPL systems, whose circuit design implements one specific digital function more efficiently than a standard architecture. The resulting circuits can be made stable and cheap. At the same time, however, this approach can at most result in consumer secure SIMPL systems, since fabricaton tolerances play no role. One particular problem is that fraudsters may try to order ASICs with the same design as the original SIMPL system at chip foundries. This may, in principle, be prevented by legal restriction.

However, a general problem is that fraudsters can order ASICs not with an identical, but with an effectively equivalent design. Since the equivalence of Boolean functions is co-NP-complete, there is no generic method for the chip foundries to notice the design equivalence. We believe that our skew design can help us to circumvent this problem, since any equivalent function seems to require the use of skew memory cells in their design. The occurence of this particular design unit can be checked relatively easily. Furthermore, it seems that skew cells have no practical applications apart from serving as SIMPL systems, whence their prohibition/restriction does not harm other implementations.

Overall, our studies have not yet matured to a level where we can claim a full successful proof of concept. But they do seem to provide an interesting first step towards the realization of electrical integrated SIMPL systems, and can also serve as a confirmation for the general feasibility of SIMPL systems as a new security tool.

Acknowledgements

The authors would like to thank Tamas Roska, Peter Vogl, Jonathan Finley, and Jürgen Schmidhuber for enjoyable and helpful discussions on the topic.

References

[1] U. Rührmair: *SIMPL Systems: On a Public-Key Variant of Physical Unclonable Functions.* Available from IACR Preprint Archive, http://eprint.iacr.org. Report 2009/255.

[2] R. Pappu, B. Recht, J. Taylor, N. Gershenfeld, *Physical One-Way Functions*, Science, vol. 297, pp. 2026-2030, 20 September 2002.

[3] R. Pappu, *Physical One-Way Functions*, PhD Thesis, MIT.

[4] Blaise Gassend, *Physical Random Functions*, MSc Thesis, MIT, 2003.

[5] Pim Tuyls, Geert Jan Schrijen, Boris Skoric, Jan van Geloven, Nynke Verhaegh, Rob Wolters *Read-Proof Hardware from Protective Coatings.* CHES 2006: 369-383

[6] G. Edward Suh, Srinivas Devadas: *Physical Unclonable Functions for Device Authentication and Secret Key Generation.* DAC 2007: 9-14

[7] P. Tuyls, B. Skoric. *Strong Authentication with PUFs*. In: Security, Privacy and Trust in Modern Data Management, M. Petkovic, W. Jonker (Eds.), Springer, 2007.

[8] P. Tuyls, B. Skoric, T. Kevenaar (Eds.) *Security with Noisy Data*. Springer 2007.

[9] U. Rührmair, J. Sölter, F. Sehnke. *On the Foundations of Physical Unclonable Functions.* Submitted, 2009. Available from http://eprint.iacr.org/

[10] C. Ellison and B. Schneier: *Ten Risks of PKI: What You're Not Being Told About Public Key Infrastructure.* Computer Security Journal, v 16, n 1, 2000, pp. 1-7.

[11] Richard P. Feynman, *Simulating Physics with Computers.* International Journal of Theoretical Physics, Vol. 21, No. 6&7, pp. 467–488, 1982.

[12] Gerald DeJean, Darko Kirovski: *RF-DNA: Radio-Frequency Certificates of Authenticity.* CHES 2007: 346-363.

[13] Srinivas B N: *SRAM for use in Physical Cryptography.* MSc Thesis, Department for Electrical Engineering and Information Technology, TU München, 2009.

[14] S. Wolfram: *Statistical mechanics of cellular automata*, Rev. Mod. Phys. 55, 601 - 644 (1983)

[15] Roska, T.; Chua, L.O., *The CNN universal machine: An analogic array computer*. Circuits and Systems II: IEEE Transactions on Analog and Digital Signal Processing, vol.40, no.3, pp.163-173, Mar 1993

[16] Kennedy, M.P.: *Three steps to chaos. II: A Chua's circuit primer*. IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, vol.40, no.10, pp.657-674, Oct 199

[17] Zou, F. and J.A. Nossek. *A chaotic attractor with cellular neural networks*. IEEE Transaction on Circuits and Systems, Vol. 38, pp. 811-812, 1991.

[18] Maciej J. Ogorzalek, Zbigniew Galias, Andrzej M. Dqbrowski, Wladyslaw R. Dqbrowski: *Chaotic Waves and Spatio-Temporal Patterns in Large Arrays of Doubly-Coupled Chua' s Circuits*. IEEE Transactions on Circuits and Systems-I: Fundamental Theory and Applications, Vol. 42, No. 10, October 1995

[19] M. Gomez-Gesteira, M. de Castro, V. Perez-Villar, L. O. Chua: *Experimental Chua's Circuit Arrays As an Autowave Simulator*, IEEE Transactions on Circuits and Systems-I: Fundamental Theory and Applications, Vol. 46, No. 4, April 1999.

[20] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer New York, 2006.

[21] L. Bottou, *Large-scale Kernel Machines.* MIT Press, 2007.

[22] L. O. Chua and T. Roska: *Cellular Neural Networks and Visual Computing: Foundations and Applications.* Cambridge University Press, 2005.

[23] A. Rodriguez-Vazquez, G. Linan-Cembrano, L. Carranza, E. Roca-Moreno, R. Carmona-Galan, F. Jimenez-Garrido,. R. Dominguez-Castro, and S. Meana: *ACE16k: The Third Generation of Mixed Signal SIMD-CNN ACE Chips Toward VSoCs*. IEEE Trans. on Circuits and Systems – I, 51(5): 851–863, 2004.

[24] L. O. Chua, T. Roska, T. Kozek, A. Zarandy: *CNN Universal Chips crank up the computing power.* Circuits and Devices Magazine, IEEE, Vol. 12, no. 4, pp. 18–28, July 1996.

[25] Cellular Wave Computers for Nano-Tera-Scale Technology – beyond spatial-temporal logic in million processor devices. Electronics Letters, April 12, 2007, Vol. 43, No. 8.

[26] Tamas Roska, private communication.

[27] S. Xavier de Souza, M. Yalcin, J. Suykens, and J. Vandewalle: *Toward CNN Chip-Specific Robustness.* IEEE Trans. on circuits and systems – I, 51(5): 892-902, 2004.

[28] D. Hillier, S. Xavier de Souza, J. Suykens, J. Vandewalle: *CNNOPT Learning CNN Dynamics and Chip-Specific Robustness.* International Workshop on Cellular Neural Networks and Their Applications, 2006.

[29] G. Csaba, X. Ju, Q. Chen, W. Porod, J. Schmidhuber, P. Lugli, U. Rührmair: *On-Chip Electric Waves: An Analog Circuit Approach to Physical Uncloneable Functions*, 2009. Available from http://eprint.iacr.org/. Report No. 2009/246.

[30] T. Roska: *Cellular Wave Computers for Brain-Like Spatial-Temporal Sensory Computing.* IEEE Circuits and Systems Magazine, Vol. 5, No. 2, pp- 5–19, 2005.

[31] Virtuoso Spectre Circuit Simulator, Cadence Design Systems, www.cadence.com.