Using different LUT paths to increase area efficiency of RO-PUFs on Altera FPGAs

Linus Feiten, Karsten Scheibler, Bernd Becker, Matthias Sauer
Chair for Computer Architecture
University of Freiburg
Georges-Koehler-Allee 51
79110 Freiburg, Germany
Email: {feiten, scheibler, becker, sauerm}@informatik.uni-freiburg.de

Abstract—Delay-based PUFs are well suited to be implemented on FPGA chips. A delay line is commonly achieved by routing the signal through a number of adjacent lookup tables (LUTs). This however uses only one of all possible routings per LUT. We suggest to implement RO-PUFs using all possible routings that can be configured at runtime, which allows to gather much more device-specific information per chip area.

Furthermore, the number of PUF response bits can be increased even further without extra hardware or software if a certain degree of correlations among the response bits is accepted. The degree of correlations is proportional to the growth of response bits, so a design decision can be made. To make this decision, a metric to quantify the degree of correlations is needed, which has so far been mostly neglected in the analyses of PUF implementations.

I. INTRODUCTION

In hardware security there are many applications in which a secret key needs to be stored securely on a device [1]; i.e. the key must not be extractable without unrealistically excessive effort. A problem with storing the key in non-volatile memory is that it might be extracted by an intrusive attack. Physically unclonable functions (PUFs, [2]) are an alternative. Instead of storing secret keys in non-volatile memory, the keys are derived from unique physical characteristics of the device. An attempt to intrusively capture the PUF signature (i.e. the device’s secret key) is likely to alter the device’s physical characteristics and with them the PUF signature itself. Furthermore, even if the PUF signature of one device could be extracted, it would be difficult to forge an identical device with the same signature, because the physical characteristics from which the device’s PUF signature is derived generally depend on uncontrollable process variations in the device’s manufacturing. It is due to this randomness that each device will ideally have its own unique PUF signature.

One way of realising a PUF in integrated circuits that has – due to its relative simplicity and stability – received much research attention, is the ring oscillator PUF (RO-PUF, [3–6]). Here, the unique physical characteristics are the timing delays of different signal paths on a chip. Figure 1 shows the circuit diagram of a ring-oscillator (RO). It consists of a single NAND-gate and a certain number of driving delay elements merely passing on the signal. When the enable input is set to logic 1, the RO starts oscillating; i.e. a constant alternation between logic 1 and logic 0 is observed at the RO’s output. An RO PUF generates one device-specific signature bit by comparing the frequencies of two ROs. Depending on which RO oscillates faster, the generated bit is 0 or 1. If the nominal delays (i.e. average expected delays over all devices) of two ROs are similar enough, the device-specific differences caused by process variations can result in either of the two ROs being the faster one with the same probability. The frequency of an RO is also influenced by temperature, voltage changes or chip ageing. However, when all ROs on a chip are uniformly affected by such effects, the result of comparing two ROs is robust against these.

Thus, two ROs are required for each PUF bit, which is very area consuming. In this work we are presenting the PDL-RO-PUF, which is an Altera FPGA implementation to yield orders of magnitude many more PUF bits from just a fraction of chip area. In fact, the method promises to yield thousands of unique uncorrelated PUF bits from the area occupied by a single RO. It takes advantage of the FPGA architecture’s lookup tables (LUTs) transforming the circular RO path into a programmable delay line (PDL). In our implementation there are \(2^{24}\) different configurations of this PDL, each resulting in a different routing of the signal through the RO. Each routing may lead to a different delay and hence RO frequency. With an automated script, we singled out pairs of PDL configurations having almost the same nominal delays while their routings still have a large enough disparity. This disparity is needed to avoid undesired correlations between PUF bits and to hamper possible modelling attacks [7].

Section II will explain the concept and implementation in more detail followed by Section III showing the results of a first simulation based on data gathered from 70 identical Altera Cyclone IV FPGAs, which are already very promising. Section IV concludes the paper and names the most pressing tasks for future work.

II. CONCEPT AND IMPLEMENTATION

A. Virtual ring oscillators (VROs)

On Altera FPGAs the ROs can be implemented using one lookup table (LUT) for the initial NAND-gate and one for
Fig. 2. From left to right we see (1) the floorplan of an Altera Cyclone IV FPGA, (2) a ring oscillator consisting of the 16 logic elements (LEs), and (3) the conceptual view of a lookup table (LUT) that is included in each LE.

Each delay element as shown in Figure 2. An \( n \)-bit LUT is a circuit that can implement any logical function with \( n \) inputs and one output. This is achieved by storing the output values for all \( 2^n \) input assignments in the LUT’s memory cells each of which is selected via multiplexers by the corresponding input assignment. LUTs are the main building blocks of FPGAs in general. The LUT shown in Figure 2 is a 4-bit LUT as provided by Altera Cyclone IV FPGAs.

To implement a delay element, the LUT simply needs to realise the identity function, which can be achieved as shown in Figure 2. If the LUT input A is used to feed in the delay element’s input, the assignments of B, C and D determine which routing the signal is actually taking through the LUT. In Figure 2 B,C,D are all assigned 1, resulting in the highlighted routing path. For a traditional RO implementation, only one routing is used which only samples the process variation of this single LUT routing. However, each of the 8 possible assignments to B,C,D would result in a more or less different routing with a potentially different process variation effect on it. The traditional RO-PUF thus only utilises a fraction of device-specific physical characteristics engraved in the chip area it uses. The PDL-RO-PUF presented here, on the other hand, is meant to tap the full potential of all individually requestable entropy sources; thus drastically reducing the chip area consumption per generated PUF bit.

Using the B,C,D inputs of a LUT to configure the routing from input A to the LUT’s output turns the LUT into an element of a programmable delay line (PDL) [8]. PDLs are traditionally implemented to dynamically adjust the delay of a line. In the context of delay based PUFs this can be used for so called Glitch PUFs [9] or to mitigate unwanted delay biases [10]. For the PDL-RO-PUF presented in this paper, we are taking advantage of the PDLs capability of sending a signal via different routings, which allows us to harvest the individual process variation not just of one but of several paths through the same LUT. The PDL’s capability to adjust the overall delay of an RO is merely utilised as a side effect. The basic idea of using different LUT routings in a PUF to yield more PUF bits has already been suggested by [11]. But there, only one path of a LUT is compared with the equivalent path of another LUT to get one PUF response bit. The increase of PUF bits is thus much smaller than that of the approach presented here (only eightfold opposed to several thousandfold). Furthermore, their design was implemented on Xilinx FPGAs, whereas our approach is tailored to the Altera architecture.

Figure 2 already showed how an RO is implemented as a row of LUTs acting as delay elements. In our PDL-RO implementation we are able to control the routing by assigning values to the B, C and D inputs of up to eight distinct LUTs. This is shown in Figure 3, not showing any further non-configurable LUTs. The limitation to eight configurable LUTs is due to routing constraints of the FPGA architecture itself which does not allow more individual inputs to the logic array block (LAB) in which the RO resides. Thus, the actual routing path through the physical RO is determined by the 24 configuration inputs conf1, ..., conf24 for which there are
2^{24} = 16,777,216 different assignments possible. Each of these assignments is realising a different routing which we call a virtual RO (VRO). The chip area hosting only one physical RO in a traditional RO-PUF implementation can thus host 16,777,216 different VROs. In the PDL-RO-PUF suggested here, one PUF bit is generated by comparing the frequencies of two VROs. If all possible VROs would be used for this, it would allow the generation of 16,777,216/2 = 8,388,608 PUF bits. However, we shall see in the following why only a certain selection of VROs should be used.

B. Sequential VRO sampling

Before we discuss the selection of VRO pairs to compare, we need to explain how the comparison of two VROs actually takes place. As the VROs manifest in the hardware of the same physical PDL-RO depending on the configuration assignment, they cannot be sampled simultaneously as it is typically the case for the RO comparison in traditional RO-PUFs.

Instead they need to be sampled one after the other. This would normally be unacceptable, because an attacker could eavesdrop the EM emissions coming from the chip to learn the sampled frequencies and thus the generated secret PUF bit. To thwart such an attack we suggest to use a true random number source to select the order in which the VROs are sampled and compared. The result is stable because the random order will be regarded both in the sampling and in the comparison. If no dedicated true random source is available on the FPGA, the least significant bit (LSB) of the last sampling’s oscillation binary counter can be used. As the counter is reset, started and stopped while the VRO is already running and the VROs are oscillating at a much higher frequency than the system clock determining the start and end of the counting, the LSB of the counter value is unpredictable even for an eavesdropping attacker. Notice though that the memory element at which the order determining bit value is stored must not be manipulable by an attacker. If an attacker is able to induce a desired value, e.g. using a fault-attack ([12], [13]), he could enforce a desired sampling order and thereby extract the PUF signature eavesdropping the oscillation frequencies. It must be hoped that fault-attacks on the overall PUF design are not possible without disturbing the normal chip functionality to such an extent that the VRO frequencies themselves will change as well. It is an assumption made for PUFs in general, that only non-volatile memory is attackable, whereas volatile memory is relatively secure.

Notice that the putative solution of having two physical ROs implementing a pair of VROs at a time for simultaneous sampling, must fail. This is due to the configurable portion of a VRO being negligibly small compared to the large non-configurable remainder of its physical host RO that remains the same for all VROs implemented in it. The process variation effects in the non-configurable parts are proportionally larger than those in the relatively small sub-paths within the configurable LUTs. Thus, the comparisons of all VRO pairs implemented in two different physical ROs will generate the same PUF bit value on a device and are therefore redundant for the device’s unique signature. The subtle differences induced by process variation into the configurable sub-paths can only be elicited by comparing two VROs of the same physical RO.

C. VRO pair selection for PUF bit generation

We shall now discuss how each VRO pair is selected in order to achieve the best results. The first prerequisite for a good VRO pair is that their nominal delays are equal. Otherwise the device-specific process variation cannot have its maximum effect and the generated PUF bit is biased. As a second prerequisite, there should not be any VROs being too similar regarding their PDL configurations. This does not only mean the two VROs of a comparison pair, but in fact all VROs of the whole PDL-RO-PUF implementation. Consider Figure 4 in which the routings of four different VROs are highlighted. In the LUTs of a physical RO, VRO a and VRO b do not share a single PDL configuration; they have the maximum possible disparity. The same applies for VRO b and VRO c. VRO a and VRO b/VRO c also do not share a single PDL configuration, however, their configurations have the same D assignments leading to a slightly decreased disparity. VRO b and VRO b/'..
have the same PDL configurations for the third and fourth LUT and the configurations for the first and second LUT only differ by the A input. Their disparity is thus minimal. Having VROs with small disparity in the VRO population brings with it a number of disadvantages.

1) Modelling attacks: Let \( f(a) \) be the sampled oscillation frequency of VRO \( a \). Let furthermore \((a < b)\) denote the sampling and comparison of VRO \( a \) and VRO \( b \), generating the PUF bit value 1 if \( f(a) < f(b) \) and 0 otherwise. According to Figure 4, let VROs \( b \) and \( b' \) be two VROs with minimal disparity, whereas \( a \) and \( b \), just like \( b' \) and \( c \), have a substantial disparity. Finally assume that \((a < b)\) and \((b' < c)\) are two samplings to generate one PUF bit each. It must be assumed that an attacker knows the PUF design and thereby knows which VRO pair is sampled at what time to generate a specific PUF bit. This could be concealed from the attacker by also randomising the order in which the PUF bits are generated, but this would require an undesirably large overhead of logic and memory circuitry in the PUF design. Only randomising the order of the two VRO samplings per PUF bit, as described above, is possible with very sparse extra circuitry and only one extra memory element. Thus the attacker knows when e.g. \((a < b)\) is sampled but he does not know if either VRO \( a \) or \( b \) is sampled first. If he eavesdrops all four VRO samplings (\((a < b)\) and \((b' < c)\), he will most likely observe one frequency of \((a < b)\) (namely \( f(b) \)) being almost identical to one of \((b' < c)\) (namely \( f(b') \)). He can thus deduce which of the two observed frequencies of \((a < b)\) is \( f(b) \) and thereby derive the resulting PUF bit depending on whether the other frequency (namely \( f(a) \)) was faster or slower. The same can be done for \((b' < c)\).

2) PUF bit correlations: Avoiding VROs of minimal disparity has another reason concerning the PUF quality. If \( b \) and \( b' \) have almost the same PDL configurations while not sharing the same amount of configurations with \( a \) or \( c \), the device-specific process variation will have pretty much the same effect on \( b \) and \( b' \), whereas \( a \) and \( c \) are affected independent thereof. As the frequencies of each two compared VROs should by design have very similar nominal delays, the nominal delays of \( a \), \( b \), \( b' \) and \( c \) are almost equal. The process variation on each specific device will determine their actual speed order. As \( b \) and \( b' \) have almost the same PDL configuration, the process variation is unlikely to produce a greater delay difference between \( f(b) \) and \( f(b') \) than between \( f(a) \), \( f(b) \approx f(b') \) and \( f(c) \). It thus only leaves the following six speed order possibilities:

<table>
<thead>
<tr>
<th>speed order</th>
<th>((a &lt; b))</th>
<th>((b' &lt; c))</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f(a) &lt; f(b) \approx f(b') &lt; f(c) )</td>
<td>1 ( \approx ) 1</td>
<td></td>
</tr>
<tr>
<td>( f(a) &lt; f(c) &lt; f(b) \approx f(b') )</td>
<td>1 ( \neq ) 0</td>
<td></td>
</tr>
<tr>
<td>( f(b) \approx f(b') &lt; f(a) &lt; f(c) )</td>
<td>0 ( \neq ) 1</td>
<td></td>
</tr>
<tr>
<td>( f(b) \approx f(b') &lt; f(c) &lt; f(a) )</td>
<td>0 ( \neq ) 1</td>
<td></td>
</tr>
<tr>
<td>( f(c) &lt; f(a) &lt; f(b) \approx f(b') )</td>
<td>1 ( \neq ) 0</td>
<td></td>
</tr>
<tr>
<td>( f(c) &lt; f(b) \approx f(b') &lt; f(a) )</td>
<td>0 ( \neq ) 0</td>
<td></td>
</tr>
</tbody>
</table>

It shows that for more than half of the possible speed orders the results of \((a < b)\) and \((b' < c)\) are inverse, indicating a correlation between the two PUF bits. For uncorrelated PUF bits, the probability of them being inverse or equal should be 0.5. Here we have a probability of 0.67 for the values being inverse. This knowledge can be used by an attacker to reduce the brute-force complexity of guessing a PUF signature, trying those bit value combinations first which due to correlations have a higher probability. To identify if such correlations exist, a metric is needed that surpasses the expressiveness of just calculating the hamming distance or bit-aliasing which is mostly done in PUF literature. We have more thoroughly elaborated on this topic and suggested a metric in [14].

One may argue that VROs with low disparity may still be used within the same comparison, because the problems addressed above only occur when they are used in different comparisons. This is correct. If \( b \) and \( b' \) are used as \((b < b')\) to generate a PUF bit, the eavesdropping attacker will not be able to determine which of the two frequencies belongs to which VRO. The generated bit will also not be correlated to other bits. Thus, there is no problem regarding PUF uniqueness. However, the other important quality measure, PUF reliability, is bound to be unacceptable because the delay difference induced by process variation between the VROs will be minuscule. If two compared VROs are too similar (not just by nominal delay but under the influence of a concrete device’s process variation), both VROs may be measured as the faster one at different sampling times. This is because the sampling process itself underlies certain uncontrollable jitter effects and the process-variation-induced difference between compared VROs must be large enough to outweigh these. The automatic algorithm described in the next section therefore dictates that all VROs used in a PDL-RO-PUF design have at least a minimum disparity to all other used VROs.

D. Automatic VRO pair coupling based on experimental data

![Assignments to LUT inputs DCB](image)

Fig. 5. Experimental results showing the average (i.e. estimated nominal) delays of all LUT configurations assignments.

If the nominal delay of each individual LUT configuration is known, the nominal delay of a VRO can be simulated according to its specific PDL configuration. In extensive experiments involving over 70 identical Altera Cyclone IV FPGAs, we have estimated the nominal delays of this architecture’s LUT configuration assignments. The necessary routing customisation has been achieved with the methods described in [15]. The results shown in Figure 5 are given to an algorithm that calculates the nominal delays of all 16,777,216 possible VROs. This only takes a few seconds on a modern CPU.
The VROs are then sorted by their calculated delay such that VROs consecutive in that order have a minimum difference of nominal delays. The algorithm traverses the stack of ordered VROs from top to bottom, building pairs for PUF bit generation removing the used VROs from the stack. If the nominal delay of the currently topmost VRO is larger than that of its successor by more than a certain delay threshold, it is discarded. This makes sure that only VROs with almost equal nominal delays are coupled. The delay threshold is given in picoseconds (ps). Furthermore, before considering the currently topmost VRO, the algorithm checks whether this VRO has a disparity lower than a certain disparity threshold with any other already used VRO. This makes sure that no two VROs with too small disparity make it into the overall VRO population. The disparity threshold is given in number of LUT configurations that must be different for any two VROs in the design.

In Figure 4 we have seen that changing the LUT configuration can have a larger or smaller impact on how different the routing within a LUT becomes. If only the B input changes, the routing difference is rather small, whereas changing the D input changes the routing more severely. The current version of our algorithm does not take this into account. It only checks if the LUTs of two VROs have the same or different configurations and increments the disparity counter by one for each difference. A future version may calculate the disparity more accurately by attributing weights according to how much the routing within each LUT actually changes.

### III. Simulation Results

The algorithm just described delivers a list of VRO pairs, each of which generates one PUF bit. Table I shows how many such pairs could be found given certain threshold values. For disparity threshold 7, the allowed combinations of LUT configurations are obviously very limited. The few possible VROs have furthermore rather large nominal delay differences, as the number of bits drops by over 30% when the delay threshold is reduced from 3.5 ps to 0.1 ps. Only the VROs of 19 pairs are at least that close together.

On the other hand, when the disparity threshold is reduced to only 4 different LUT configurations, there are 8191 viable VRO pairs whose nominal values are already very similar. Reducing the delay threshold here only discards three pairs, as all the others have already been closer together than 0.1 ps before. Recall that this amount of PUF bits can actually be generated with a chip area not even capable of generating a single bit in the traditional RO-PUF!

<table>
<thead>
<tr>
<th>Table I. Number of PUF bits for different thresholds.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delay Threshold</td>
</tr>
<tr>
<td>3.5 ps</td>
</tr>
<tr>
<td>0.1 ps</td>
</tr>
</tbody>
</table>

The final lists of VRO pairs need to be implemented and tested on the actual Altera Cyclone IV FPGAs, to see if the generated PUF bits actually fulfill the requirements of a competitive PUF implementation. Presenting such results will be part of a future publication. For this work, we took the samplings of the initial experiments we already used to calculate the nominal delays of every LUT configuration. The raw data includes the measured delays of all individual FPGAs, so we can use these to simulate the VRO frequencies on each individual FPGA and thereby the PUF bits of each FPGA.

Figure 6 shows the results of the normalised bit-aliasing metric (cf. [10], [14]) for each implementation variant. Each bit gets a value between 1.0 (perfect) and 0.0 (poor). In the graph the bits are sorted from left to right by their bit-aliasing values to get a good visual impression of how many good, bad and average bits there are. 1.0 is the perfect value, meaning that for half of the devices the bit was 1 and for the other half 0. The value 0.0 means that the corresponding bit had the same value (either 1 or 0) on all devices. If an attacker trying to guess a device’s PUF signature knows about such biased bits, he can reduce the brute-force effort by trying the more likely values first. For all our PDL-RO-PUF designs, however, we can attest that the bit-aliasing results are close to perfect. Lowering the delay threshold does in fact remove the worst bits, because when the nominal delay difference of a VRO pair is too large, the device-specific process variation might often not come into full effect.

To determine the PUF uniqueness, however, it is not enough to just calculate the bit-aliasing metric\(^\text{1}\), as it only regards each bit by itself. Correlations among bits are not

\(^{1}\)Let alone the hamming-distance metric that is in fact just another, more complicated way of expressing bit-aliasing (cf. [14]).
grasped with this metric. In [14] we have suggested a metric capable of doing so. Figure 7 shows its result plots for the PDL-RO-PUF designs with delay threshold 0.1 ps, which are to be interpreted as follows.

The y-values stand for the degree of correlation of any two PUF bits; i.e. how often did two bits assume the same or the inverse value over all devices. −1.0 means that two bits have been inverse on all devices. 0.0 means they have been equal on all devices. Huge correlations like that render at least one of the bits redundant because it does not hold any unique information. A y-value of 0.0 means that two bits have just as often been observed being inverse as being equal. The x-values stand for how many PUF bit pairs have a particular degree of correlation. E.g., a blue dot at (1.0, 5%) would mean that 5% of all possible PUF bit pairings were equal on all devices. For n PUF bits, every bit is checked with every other bit, so there are n(n−1)/2 many pairings. To see what a perfectly uncorrelated distribution would be, the ideal result is plotted in yellow.

A peculiar phenomenon for disparity 4 are 192 bit pairings (out of all 33,517,578 pairings) with a correlation value of 1.0, indicated by the blue dot at the bottom right. Investigations of the raw data showed how these come to pass. Consider the LUT configurations below and let one of the correlated bits k be generated from (VRO1 < VRO2) and the other bit l from (VRO11 < VRO12). Now k1 and k2 (highlighted red) and l1 and l2 (highlighted blue) have each four of their LUTs configured the same, which is not violating the disparity 4 rule. Likewise, the remaining four configurations are the same for k1 and l1 (yellow) and for k2 and l2 (green). Thus, every time k1 is actually faster than k2 on a device, l1 must also be faster than l2 and the generated PUF bits are always the same. As long as the number of such bits is still in the parts per thousands range like here, it does not do much harm. In future work, however, we will consider ways of enabling the VRO selection algorithm to exclude such redundancies.

![Plots of the PUF bit correlation metric for different PDL-RO-PUF implementations](image)

Finally, we have to talk about PUF reliability. As we have sampled each LUT configuration ten times, we were able to simulate how stable the PDL-RO-PUF responses can be expected to be when also sampled ten times. Table II shows the percentages of bits per design that were sampled as 1 with a certain ratio. 1.0 means that out of the ten samplings the bit was always sampled as 1. 0.9 stands for nine out of ten, until 0.0 stands for a bit being sampled ten out of ten times at 0.

<table>
<thead>
<tr>
<th>Ratio of a bit being 1</th>
<th>Disparity Threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>7</td>
</tr>
<tr>
<td>1.00</td>
<td>43.27%</td>
</tr>
<tr>
<td>0.9 – 0.7</td>
<td>42.35%</td>
</tr>
<tr>
<td>0.6 – 0.4</td>
<td>42.22%</td>
</tr>
<tr>
<td>0.3 – 0.1</td>
<td>41.31%</td>
</tr>
</tbody>
</table>

We separated the bit ratios in five groups. 1.0 and 0.0 are both perfect reliability. But 0.9 – 0.7 and 0.3 – 0.1 are also still manageable without having to use error correction. When the PUF signature is created, each VRO and hence each bit can be sampled several times in a row after which a majority voter determines the dominant value. Problematic are only those bits that do not even have a strong tendency towards one or the other value. For these bits, error correction is required, if a stable PUF response is needed [16]. 5% potentially erroneous bits could still be handled by an error correcting code. It must also be assumed that the values given here are too pessimistic, as they are based on only 10 samples. Our upcoming experiments, in which the VROs will be sampled with their actual configurations instead of just being simulated based on prerecorded data, will include a larger sample size regarding reliability. This will also be the time to discuss the effects of decreased disparity on PUF reliability.
IV. CONCLUSION AND FUTURE WORK

While the real field experiments with the PDL-RO-PUF are still to be carried out, this paper has introduced the concept and shown evidence based on previously sampled data that the PDL-RO-PUF could in fact be viable. If this should substantiate, it will be a major step for delay based PUFs on FPGAs, as the number of response bits generated per chip area could be increased tremendously.

A major challenge still unsolved is how the PDL configuration bits (24 bits per VRO) are to be stored or generated. Just storing them is probably not a viable option, as the required space of tamper-resistant memory would be immense. The tamper-resistance of the PDL configurations is important, because if an attacker can manipulate them, he can generate exactly the configurations he needs to mount a modelling attack as described above. But then again, the whole purpose of having a PUF on a chip is that there is no trusted non-volatile memory available.

We are at the moment thinking in different directions. Maybe there are reoccurring patterns in the PDL configurations that could be programmed into a finite state machine running on the FPGA. This would be far more complicated for an attacker to manipulate. Or we will look at solutions from the field of test vector compaction where the problem has been tackled for a long time of how to generate a desired set of bit strings without having to explicitly store them. A linear-feedback shift register (LFSR) could be deployed to generate at least a certain subset of PDL configurations.

ACKNOWLEDGEMENTS

Sincere thanks are owed to Michael Pehl and his colleagues from the Chair of Security in Information Technology at the Technical University of Munich (TUM) for fruitful discussions and advice.

REFERENCES