FPGA Implementation of a Statistical Saturation Attack against PRESENT

Stéphanie Kerckhof, Baudoin Collard, François-Xavier Standaert

UCL Crypto Group, Université catholique de Louvain.
Place du Levant 3, B-1348, Louvain-la-Neuve, Belgium.
e-mails:stephanie.kerckhof; baudoin.collard; fstandae@uclouvain.be

Abstract. Statistical attacks against block ciphers usually exploit "characteristics". A characteristic essentially defines a relation between (parts of) the block cipher's inputs, outputs and intermediate values. Intuitively, a good characteristic is one for which the relation between the cipher's inputs and outputs exhibit a significant deviation from the uniform distribution. Due to its intensive computational complexity, the search for good characteristics generally relies on heuristics, e.g. based on a branch-and-bound algorithm. But the use of such heuristics directly raises the question whether these good characteristics remain good, as the number of cipher rounds increases. This question relates to the so-called hull effect, expressing the idea that in a practically secure cipher, only the combination of many characteristics can explain the statistical deviations exploited in cryptanalysis. As characteristics are also a central tool when estimating the data complexities of statistical attacks, determining whether a hull effect can be observed is essential in the security evaluation of a block cipher. Unfortunately, this is again a computationally intensive task, as it ideally requires to sample over the full input space. In this paper, we consequently discuss the interest of hardware assistance, in order to improve the understanding of statistical attacks against block ciphers. More precisely, we propose an FPGA design that allowed us to evaluate a statistical saturation attack against the block cipher PRESENT, for overall complexities up to 2^{50}. Compared to previous software solutions, it corresponds to an increase of the maximum data complexity experimentally reached up to now by a factor 2^{14}. Our experiments confirm that up to 19 rounds of PRESENT can be broken with 2^{48} plaintext/ciphertext pairs. They also serve as a basis for discussing the statistical hull effect and suggest that 31-round PRESENT should be safe against such statistical attacks.

1 Introduction

Since its publication in 2007, PRESENT has been one of the most carefully investigated low cost ciphers. Several papers have analyzed its security against different types of cryptanalysis. Starting in 2008, Wang presented a differential cryptanalysis of reduced round PRESENT, allowing one to attack 16 rounds (out of 31), with 2^{64} chosen plaintexts [23] (these results have been recently rediscussed in an IACR ePrint report [15]). The same year, Z'aba et al. presented a bit-pattern integral attack that was able to break up to 7 rounds of PRESENT-128 (the 128-bit key version of the cipher), with 2^{24} chosen plaintexts, and a significant time complexity of 2^{100} partial decryptions [25]. This paper extended previous works on square (aka integral, aka saturation) attacks to ciphers with bit-oriented transforms. Different additional results appeared in 2009. In [19], Nakahara et al. analyzed the security of PRESENT-128 against attacks based on the linear hull effect, claiming to break 25-rounds of PRESENT-128 with the full codebook (again with a time complexity of approximately 2^{100}). They also experimented a purely algebraic attack able to break 5 rounds of PRESENT with 5 known plaintexts (and a few minutes of offline computations). In parallel, Ohkuma presented another linear attack against 24 rounds of PRESENT (80-bit version), with the full codebook, taking advantage of the linear hull effect for a certain class of weak keys [21]. Related-key cryptanalysis of PRESENT was additionally investigated in [22], for 17 rounds. And in a paper from FSE 2009 [1], dedicated to the combination of algebraic and differential cryptanalysis, Albrecht and Cid proposed various attacks against reduced versions of PRESENT. For example, they described a 16-round attack with complexities similar to the ones in [23]. More recently, Cho proposed a multidimensional linear attack, claiming to recover the 80-bit secret key of PRESENT for 25 rounds, with the full codebook [5]. Different empirical evaluations of reduced-round variants (with 6,7,8,9 rounds) were proposed in the paper, allowing to put forward the interest of the multidimensional approach. One can also mention the experiments of Blondeau and Gérard [3], used to confirm their theoretical analysis of differential cryptanalysis. Finally, the statistical saturation attack we experiment in this work has been introduced in [6] and then extended to multiple trails at ACNS 2010 [7].

As usual in cryptanalysis, one limitation shared by most of these previous works is that their estimated data complexity strongly relies on assumptions that may not be fulfilled, as the number of rounds in a block cipher increases. For example, security evaluations against linear cryptanalyses usually exploit Matsui's piling up lemma [18], that simply multiplies the linear biases of single-round linear approximations. A straightforward application of the lemma leads to the counter-intuition that increasing the number of rounds in a cipher may arbitrarily increase its security against linear attacks (as the bias can then be arbitrarily close to zero). In fact, as first explained by Nyberg in 1994 [20], correct estimations of the data complexity in a block cipher require to consider linear hulls (i.e. sets of linear characteristics sharing the same input/output masks). Yet, in practice, the number of characteristics in a hull increases exponentially with the number of rounds, and is rapidly impossible to exploit. Hence, present cipher designs, such as the AES Rijndael, are frequently based on the paradigm of practical security. That is, one assumes that a cipher is secure against linear cryptanalysis if the data complexity determined from the best characteristic in a cipher is prohibitive [14]. And excepted for the investigations of Keliher et al. in [12, 13], and the investigations of small scale variants of block ciphers, in [8], few experimental works tackled the problem of determining how many block cipher rounds are actually needed for the linear hull effect to be significant.

In other words, most experiments against real world ciphers consider number of rounds for which the statistical deviations can still be explained by one (or few) characteristics. Such a limitation is typically exemplified by the statistical saturation attack against PRESENT, that can be viewed as a particular case of the multidimensional cryptanalysis described in [11] (see the recent work of Leander [17]). Hence, it is natural to question the validity of the data complexity estimations for large number of rounds, as given in [6].

In this paper, we consequently provide two contributions to the cryptanalysis of the block cipher PRESENT. First, starting from the observation that experimental validation is still a useful step for increasing our understanding of statistical attacks, we investigate the computational power that can be gained by outsourcing parts of the computations to a dedicated hardware platform. For this purpose, we developed a hardware-software co-design, based on an FPGA board, allowing us to accelerate the most consuming tasks of a statistical saturation attack against PRESENT, while keeping the communication rate between the different parts of the system reasonable. We note that the design is generic, and could easily be modified to investigate similar attacks, e.g. linear or differential. Our results include an investigation of different implementation tradeoffs and technologies, together with one fully functional prototype, based on a Xilinx Virtex-5 device. Second, we used our co-design to launch large-scale experimental attacks against 15,16,17 and 18-round PRESENT, with data complexities of up to 2^{48} per attack. These experiments confirmed the previous analyzes from [6], i.e. a data complexity increase by a factor of 2^3 per round, for up to 18 rounds of PRESENT (up to 19 rounds, if a two-round partial decryption is used). By providing a careful investigation of the statistical distributions exploited in the attack, and their key-dependent behavior, our results also allow discussing the apparition of a statistical hull effect in PRESENT on a concrete basis. They suggest that 31 rounds of PRESENT should be safe against statistical attacks. We finally conclude the paper by proposing directions for better selecting the number of rounds in a block cipher.

2 Background

2.1 The block cipher PRESENT

PRESENT is an ultra-lightweight block cipher designed for hardware constrained environments, such as RFID tags and sensor networks. It is a 31-round SPN (Substitution Permutation Network), and it was introduced by Bogdanov et al. at CHES 2007 [4]. The block length is 64 bits and the possible key lengths are 80 and 128 bits. Each of the 31 rounds is composed of a XOR operation, a nonlinear substitution layer and a permutation layer, operating as follows. First, the 64-bit input of the round is XORed with the round subkey. The result of that operation is then passed through the substitution layer, which consists of 16 identical 4x4 S-boxes applied in parallel. Finally, the permutation layer performs a bit-by-bit permutation.

2.2 Statistical Saturation Attacks

The statistical saturation attack, originally described in [6], is based on a weakness in the diffusion layer of PRESENT. This weakness can be observed in Figure 1, where it is shown that only 8 out of the 16 output bits of S-boxes 5, 6, 9 and 10 are directed to other S-boxes. Hence, by fixing certain plaintext bits, we are able to observe a non-uniform distribution at the output of the round. Since the input and output bits of the bold trail highlighted in the figure are the same, it is then possible to iterate this weakness for several rounds. In order to turn this weakness into a key recovery attack, one finally assumes that the distribution at the output of the trail remains significantly different from uniform as the number of rounds increases. Hence, by doing a partial decryption through the last encryption round, one can select the key candidate that maximizes the Euclidean distance between the experimental distributions obtained for all the key candidates and the uniform one. If the attack is successful, the key maximizing this distance is the correct one. In the following of the paper, we

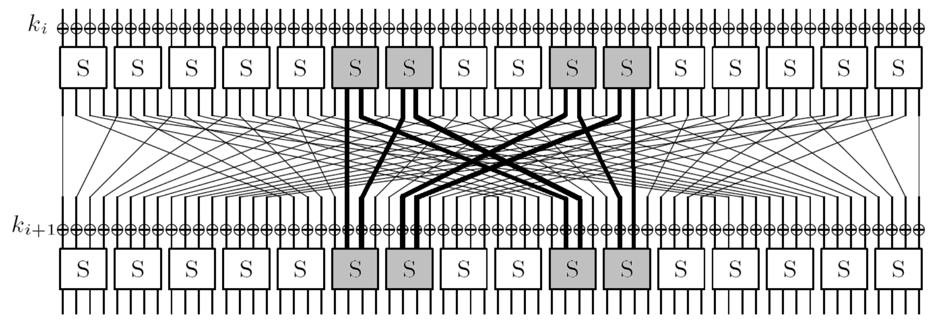


Fig. 1. Weakness of the diffusion property of the PRESENT

focus ourselves on two variations of the basic attack, denoted as Extension 1 and Extension 2 in [6]. First, we enlarged the fixed part of the plaintext to 32 bits, in order to increase the non-uniformness of the target distributions. Second, we performed the analysis multiple times, using different values for the 32-bit fixed part of the plaintexts. In other words, we carried out many sub-attacks obtained from sets of 2^{32} varying plaintexts, for different fixed input patterns. Then, for each key candidate, we re-combined the results, by simply taking the sum of the uniform vs. measured distances given by these different 32-bit sub-attacks.

3 Hardware Architecture

As described in the previous section, a statistical saturation attack is composed of three phases. First, a large number of plaintexts are encrypted and the corresponding ciphertexts are collected. Then, a distribution is computed from the resulting ciphertexts. Finally, given this experimental distribution, a partial decryption is processed and the resulting $R-1$-round distributions are tested w.r.t. uniform. From these three phases, the encryption part is the most time consuming one. On the contrary, the time needed by the partial decryption is not really critical. Therefore, we first decided to implement the PRESENT encryption in hardware, letting the partial decryption task to a software. Next, regarding the distribution generation, we also chose to implement it in hardware, for data rate reasons. As will be clear in section 3.2, our implementation of PRESENT has a huge output bitrate. Implementing the distribution generation in hardware allows us to reduce the output bitrate of our FPGA by a factor of 2^{24} (see section 3.3). Note finally that we did not implement the key schedule in hardware. The keys are generated by a software and provided to the FPGA by an Ethernet port. In the remaining of the section, we will first describe the FPGA technology we used for our implementations. We will then detail the architecture choices we made for the PRESENT encryption and distribution generation. We conclude the section with an overview of the complete system and a description of its performances.

3.1 Hardware Technology

The technologies we used to implement our architecture are Virtex-5 [24] and Virtex-6 FPGAs from Xilinx. The main logic resources of those FPGAs are the CLBs (Configurable Logic Bloc). Those CLBs are divided into two slices which are themselves composed of four logic-function generators (or look-up tables), four storage elements, wide-function multiplexers, and carry logic. These elements are used by all slices to provide logic, arithmetic, and ROM functions. In addition to this, some slices support two additional functions: storing data using distributed RAM and shifting data with 32-bit registers. Slices that support these additional functions are called SLICEM; others are called SLICEL. Figure 2 illustrates a SLICEL. The function generators in Virtex-5 FPGAs are implemented as 6-input look-up tables (LUTs). Each LUT possess 6 independent inputs (A1 to A6) and 2 independent outputs (O5 and O6). It can either implement any arbitrarily defined six-input Boolean function (only O6 is used in this case) or two arbitrarily defined five-input Boolean functions, as long as these two functions share common inputs (both O5 and O6 are used in this case). Signals from the function generators can exit the slice (through A, B, C, D output for O6 or AMUX, BMUX, CMUX, DMUX output for O5), enter the XOR dedicated gate from an O6 output, enter the carry-logic chain from an O5 output, enter the

select line of the carry-logic multiplexer from O6 output, feed the D input of the storage element, or go to F7 multiplexers from O6 output. Slices also contain three multiplexers (F7 and F8) that can be used to combine up to four function generators and provide any function of seven or eight inputs in a slice. The storage elements in a slice can be configured as either edge-triggered D-type flip-flops or level-sensitive latches. The D input can be driven directly by a LUT output or by the AX, BX, CX, or DX slice inputs bypassing the function

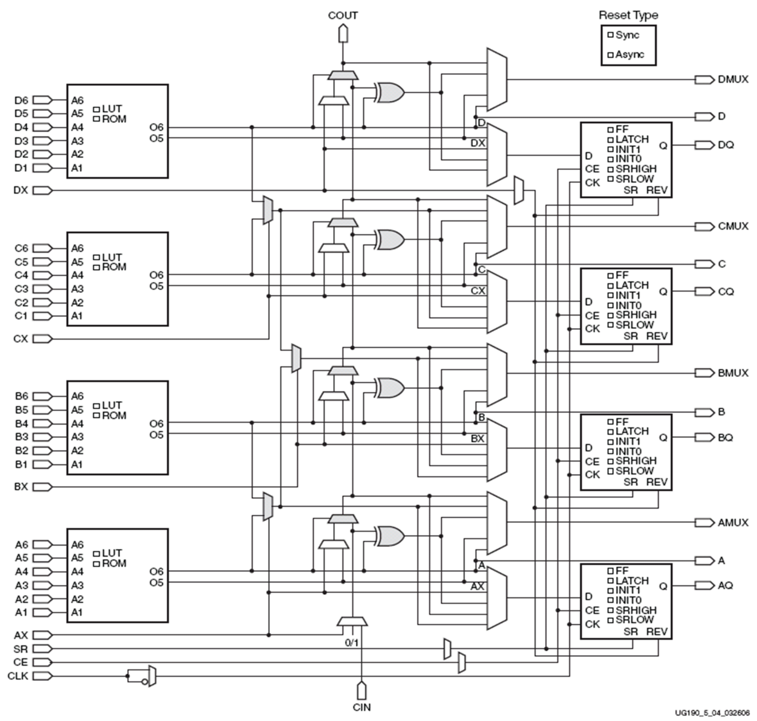


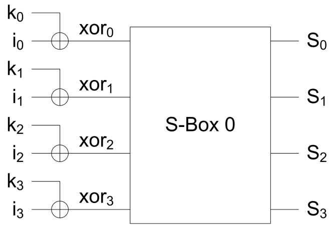
Fig. 2. Diagram of a SLICEL

generators. The slices composing a Virtex-6 FPGA are quite similar to those of a Virtex-5. The major difference comes from the possibility to register both LUTs outputs (O5 and O6) in separate flip-flops. Finally, in addition to distributed RAM, Virtex-5 and -6 FPGAs include a large number of 36 Kb block RAMs. Each 36 Kb block RAM contains two independently controlled 18 Kb RAMs.

3.2 PRESENT Architecture

PRESENT was originally designed to be extremely low cost and easy to implement in hardware. Therefore, the resources needed by a round of PRESENT are quite limited. In this section we will focus on the XOR and S-boxes layers, the permutation layer resulting only in routing which is not resource consuming from an FPGA implementation point of view. In order to determine the resources needed by a round of PRESENT, we first detail the resources consumed by the smallest relevant part of a round. This corresponds to one S-box and its corresponding XORed inputs, as depicted in left part of Figure 3.

As previously described, Virtex-5 FPGAs are based on slices composed of four 6-bit LUTs and four 1-bit registers. Therefore, an optimal way to reduce the LUTs used is to regroup all the logical operations in order to obtain a minimum number of blocks that take 6-bit inputs and give 1-bit outputs. Furthermore, in order to be speed efficient, it is also recommended to limit the number of logic



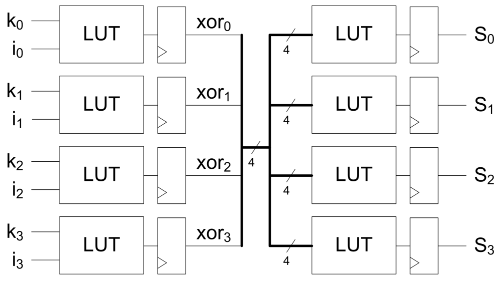
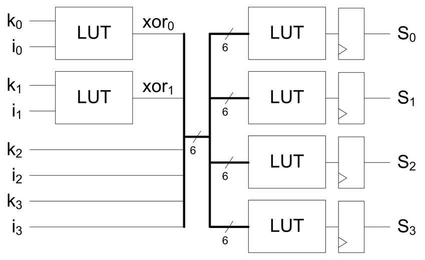


Fig. 3. Diagram of an S-box with its XORed inputs (left), equivalent LUT representation (right)

levels between two registers (a logic level corresponding to one LUT). The first possible way of implementing the combination of one S-box and four XOR operations in hardware is to limit our architecture to only one level of logic between two registers. Here, one LUT is needed per XOR operation and four LUTs are needed for the S-box (see right part of Figure 3). Even if, this architecture is the most speed efficient, it is also the most resource consuming.



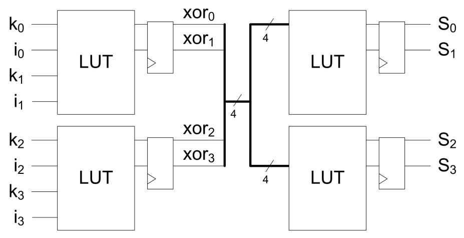


Fig. 4. VIRTEX-5 (left) and VIRTEX-6 (right) LUT representations of a round part

A second possibility is to combine some of the XOR operations with the S-box, as illustrated in left part of Figure 4. The number of LUTs needed by this architecture is decreased by two in comparison with the previous one. However, it is now composed of two levels of logic between two consecutive registers.

Finally, Virtex-6 FPGAs have two times more registers than Virtex-5 ones. This gives the possibility to store the two outputs of each LUT. Therefore, to reduce the number of used LUTs, we now need to regroup all logic to either form blocks that takes 6-bit input and 1-bit output, or blocks sharing 5 identical input bits, with 2-bit output. As illustrated in the right part of Figure 4, the number of needed LUTs to implement an S-box and four bitwise XOR is now only four. We indeed combined two XOR operations in a single LUT and the S-box in two of them.

To implement a round of PRESENT, the previous blocks must be repeated 16 times. 64-bit key registers are also needed for each round. In order to be speed efficient, we decided to fully unroll PRESENT, which allows us to encrypt a new plaintext every clock cycle. The implementation results for a 32-round PRESENT are given in Table 1, where V5 - 64 stages is the Virtex-5 design having two levels of registers per round, and V5 - 32 stages is the design having only one such level. These results confirm that the Virtex-5 architecture with 32 stages needs almost 2000 registers less than than the one with 64 stages. The maximum frequency also decreases from one architecture to the other. However, the frequency that can be reached by the 32-stages architecture is more than sufficient as we will later choose to run our complete design at a frequency of 125 MHz (see section 3.4), corresponding to a bitrate of 7,5 Gbps. Results obtained with Virtex-6 FPGAs are even better than those obtained with Virtex-5. The number of slices needed with Virtex-6 is almost half the one needed by the V5 - 64 stages architecture, while the maximum frequency is the same. However, the board on which the experimental tests were performed is a Virtex-5 one and we have therefore decided to use the V5 - 32 stages architecture for our design.

	V5 - 64 stages	V5 - 32 stages	V6 - 64 stages
LUTs	4077	3086	3130
Registers	6177	4162	6177
Slices	1561	1420	819
Max. Frequency	588 MHz	470 MHz	588 MHz
Bitrate	35 Gbps	28 Gbps	35 Gbps

Table 1. Implementation characteristics of 32-rounds PRESENT

Note finally that the bitrate reachable with these different architectures is anyway far too high to be output by our FPGA interfaces. As previously said, a solution to avoid this interface issue is to compute the distributions on board in order to reduce the data rates, as it will be explained in next section.

3.3 Distribution Generator Architecture

The statistical saturation attack exploits the experimental distributions of a few chosen ciphertext bits. In particular, the trail in Figure 1 involves 16 output bits of which the distribution has to be partially decrypted. In order to decrease the size of the distributions to store in our FPGA implementation, our experiments are based on the analysis of two 8-bit distributions, corresponding to the output of S-boxes 5 and 9 for the first one, and S-boxes 6 and 10 for the second one. This is possible because the partial decryption needed in the key recovery phase can be applied independently for the two sets of S-boxes. A distribution generator was then used to compute those two distributions. Half of this generator is illustrated in Figure 5. It is composed of Virtex RAM blocks of 18 kilobits, an adder, different multiplexers, some additional logic, and essentially works as follows.

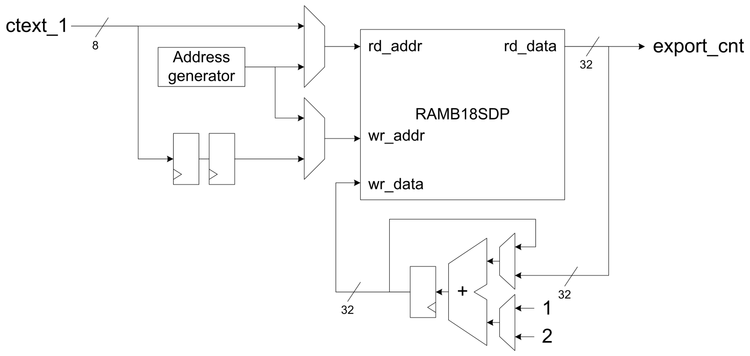


Fig. 5. Hardware architecture of half of the distribution generator

We first need 256 counters to compute each distribution and, because we chose to implement Extension 1 of the statistical saturation attack, the distribution has to be computed on 2^{32} ciphertexts, which corresponds to at most 32 bits per counter. Those counters are saved in a Virtex RAM block and are loaded by using the 8-bit ciphertext value as a RAM address. The loaded counter is then incremented and written back in memory. This whole process takes three clock cycles to be performed from the moment a ciphertext is available as RAM address to the moment the counter has been updated in RAM. However, to have a continuous flow between PRESENT and the distribution computation, we must be able to update the counters every clock cycle, which means that if at least two out of three consecutive ciphertexts values are identical, the counter in RAM must still be updated properly. For this reason, we added some logic before the adder, which gives us the opportunity to choose between the RAM output and the last incremented counter.

The 256 32-bit counters are exported once every 2^{32} clock cycles which corresponds to a decrease of PRESENT's output bitrate by a factor of 2^{24}. To avoid a loss of time during the exportation, we allocated two RAM blocks per distribution so that the second RAM is used for the computation of a new distribution while the first is being emptied and reset.

The implementation results for the complete distribution generators (composed of two of the illustrated parts) are given in Table 2. The maximum reachable frequency with the distribution generator is lower than the one we had with PRESENT. This is due to a higher number of logic levels between two registers. Indeed, we wanted to limit as much as possible the number of cycles needed to update a counter resulting in longer critical paths.

3.4 Complete Design

The complete design has been implemented on a Xilinx XUPV5 board from which we used the Ethernet port to communicate with a computer. It is illustrated in Figure 6 and works as follows. First, controls such as round key values,

	Virtex-5	Virtex-6
LUTs	519	651
Registers	332	269
Slices	205	193
RAM (18 kb)	4	4
Max. Frequency	232 MHz	205 MHz

Table 2. Implementation characteristics of two 8-bit distribution computation

destination MAC address and plaintext initial values are sent to the FPGA board through the Ethernet port. The received Ethernet packets are processed by the Ethernet Media Access Controller (MAC) and sent to an 8-bit width FIFO. A packet parser parallelizes the FIFO's output and sends the relevant information to the statistical saturation attack (SSA) block. The SSA block encrypts a large number of plaintexts and computes the corresponding distributions. The distributions are then sent to the packet builder in order to form Ethernet packets which are finally sent on an Ethernet link.

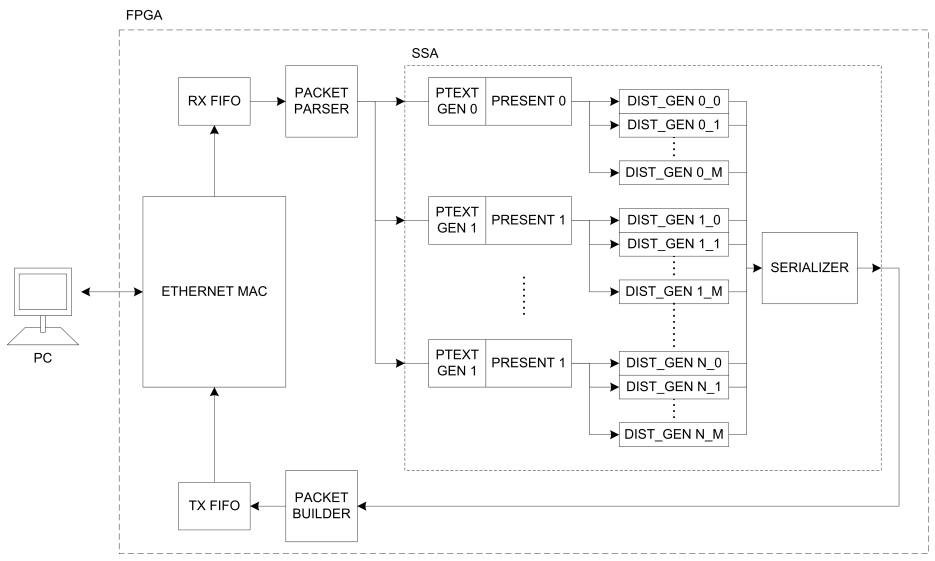


Fig. 6. Block diagram of the complete design

The FPGA available on our board is a Virtex-5 LX110T FPGA. The number of PRESENT blocks those FPGAs can contain depends on the number of rounds implemented per PRESENT block. For our experimentations, we used 18-round PRESENT and, in order to obtain more experimental results, we computed the distributions for four different rounds simultaneously. With this configuration, we would be able to fit up to 16 PRESENT blocks, and the 64 corresponding distribution generators in a single FPGA. However, due to timing problems during the synthesis of such a huge design, we decided to limit the final implementation to 8 PRESENT blocks and 32 distribution generators. We also decided to have an identical clock frequency for the complete design, which is the same as the one needed by the Ethernet MAC: 125 MHz. At that frequency, the complete design encrypts more than 2^{29} plaintexts per second and outputs 64 distributions (8 per PRESENT block) every 34 seconds.

4 Experimental Results

In this section, we take advantage of the previously described design in order to launch large scale experimental attacks against PRESENT. The goal of these experiments is twofold. First, we aim to challenge the theoretical data complexity estimations of the statistical saturation attack given in [6]. In particular, under some independence assumptions detailed in this previous work, it is expected that the data complexity of an attack exploiting the bold trail in Figure 1 increases by a factor of 2^3 per round. But as for linear cryptanalysis, this estimation should become incorrect as soon as a statistical hull effect starts to have a significant impact on the distributions of the ciphertexts. Next, we note that although the use of an FPGA board allows us to gain a significant computing power compared to previous software-based experiments, our results are still limited. Namely, we performed 5 attacks against 5 independent keys, and each of these attacks was bounded to a data complexity of 2^{48} (which is still far away from the codebook). These limitations are naturally justified by time constraints: each of our 5 attacks corresponds to 3.5 days of computations. It implies a limited sampling, both in terms of keys and plaintexts, that has to be considered in the interpretation of the results. Hence, we aim to take advantage of our experiments to discuss the hull effect in general, and whether it can be detected by experimentally sampling only a part of the plaintext space.

In the following, for each of the two 8-bit distributions exploited in the attack, we consider two main evaluation metrics. We first estimate the gain[1]. That is, if an attack is used to recover an n-bit key and is expected to return the correct key after having checked on the average M candidates, then the gain of the attack, expressed in bits, is defined as:

$$\lambda = -\log_2 \frac{2 \cdot M - 1}{2^n} \tag{1}$$

We provide gains averaged over the 5 experimented keys, for the two 8-bit distributions taken independently (in the left part of Figure 7), and their average (in the right part of the figure). Next, we provide estimates for the Euclidean

[1] Alternative metrics, such as the advantage used by Gérard and Tillich in [10], would allow deriving additional insights on the performances of the attacks, but are harder to estimate in view of our very limited sampling.

distance between the partially decrypted output distributions and the uniform distributions. Distances are computed for the correct key candidate, and averaged for all the wrong key candidates, hence allowing to observe if the correct key candidate can be easily distinguished. These distances are again averaged over our 5 experiments. We also plot these distances for each tested key independently, in order to exhibit how their variance compares to the previous mean values. This second metric, computed for data complexities from 2^{32} to 2^{48}, and number of rounds from 15 to 18, is given in Figures 8, 9, 10, 11. We now detail some important observations that can be derived from these plots.

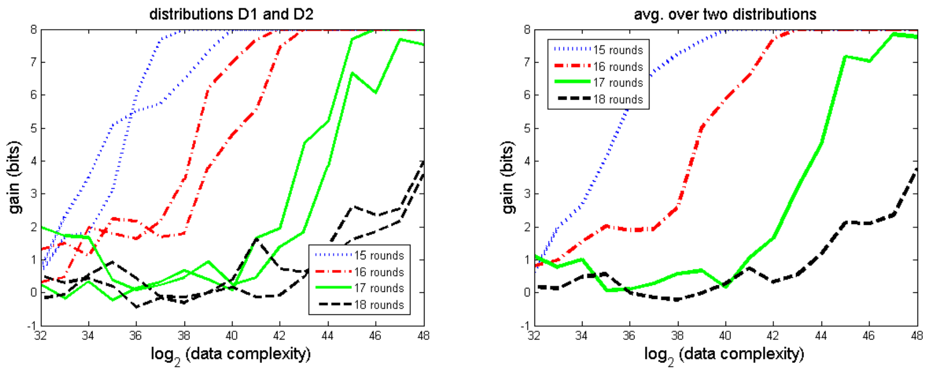


Fig. 7. Gains of the attacks.

First, regarding the gain pictures (Figure 7), one can see that the 2^3 multiplicative factor is quite accurately observed for up to 17 rounds. We also remark that the two investigated distributions do not behave exactly in the same way (this will be confirmed by the distance to uniform metric). As for the 18th round, a non-negligible gain can still be observed, but more sampling data would be required to analyze this setup with more confidence.

Regarding the distance to uniform metric, we again analyzed the combination of the two distributions (in Figure 8) and these distributions taken separately (see Figures 10 and 11, in appendix). The general observation, also confirming theoretical predictions, is that the distance between the average behavior of the correct key candidate and the average behavior of the wrong key candidates decreases with the number of rounds. In addition, we plotted this metric for the five correct key candidates of our experiments on the figures, for data complexities between 2^{42} and 2^{48} (with blues crosses for distribution D1 and red circles for distribution D2). One can notice that the scattering of these good key candidates becomes more important compared to the distance between the average curves on the plot, as the number of rounds increases. In other words, the problem of

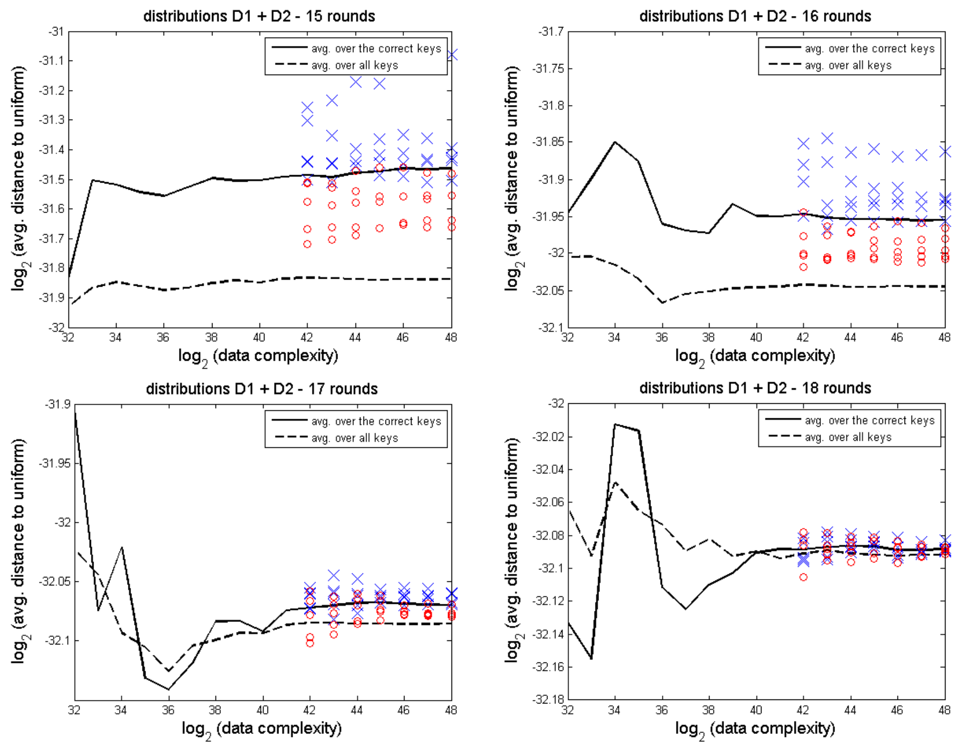
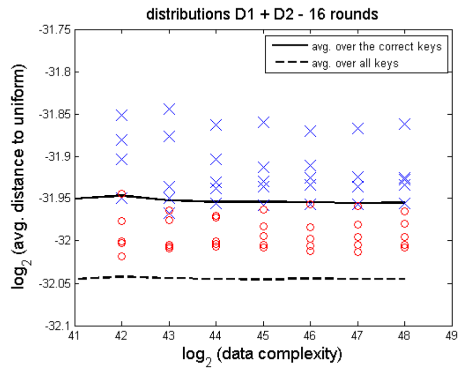


Fig. 8. Distance to uniform of the two distributions.

recovering the keys by distinguishing these distributions becomes more difficult. For round 18, this scattering even encompasses the two average curves[2].

Eventually, the central question behind these experiments is to know whether these plots indicate the apparition of a non-negligible hull effect for round 18. In other words, is the closeness between the correct and wrong key candidates due to such an effect or is it caused by a too small data complexity (the theoretical data complexity for attacking 18 rounds is 2^{51})? For answering this question, it is most interesting to observe the zoomed pictures of Figure 9. On the left part of the figure (i.e. for round 16), one can clearly see that the distributions D1 and D2 can be distinguished for all key candidates - even before the theoretical data complexity of 2^{45} is reached (this can be further observed from Figures 10 and 11 in appendix). By contrast, in the right part of the figure (i.e. for round 17), there is a significant overlap between the two distributions - in particular

[2] The average value of the distance to uniform metric is close to 2^{32}, independent of the number of rounds. This directly relates to the use of Extension 2 in our experiments. That is, we evaluate the combination of several sub-attacks of data complexity 2^{32}, where the combination of sub-attacks is performed by a (heuristic) sum of the average distances.



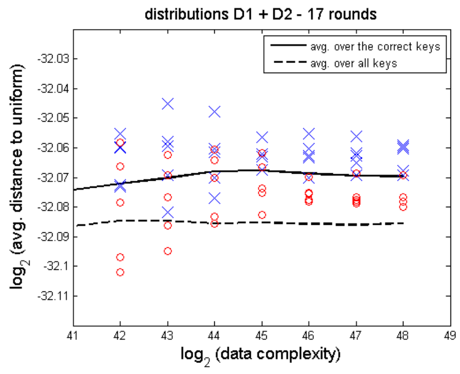


Fig. 9. Distance to uniform of the two distributions (zoom).

when the theoretical data complexity of 2^{48} is not reached. Referring to the small scale experiments in [8], this plot consequently suggests the apparition of a statistical hull effect, with distributions that become harder to distinguish and key dependent. We note again that these observations have to be taken with care, as they are based on visual inspection and not backed up with sufficient statistical confidence (again, due to the computationally intensive nature of our experiments).

5 Conclusion & Open Problems

This paper first highlights the interest of recent reconfigurable devices (FPGAs) in the context of statistical cryptanalysis. Such hardware assistance allowed us increasing the experimental data complexities reached in previous experiments[3], by a factor of 2^{14}. These important gains are due to the very convenient setting of most statistical cryptanalyses, in which one needs huge computing powers, with limited connectivity between the hardware and software parts of the system. In this respect, the design proposed in this paper could possibly be improved to gain some (small) additional factors. Focusing our design on only one or two target rounds (rather than four in the present case) and moving to the more recent Virtex-6 technology are typical examples of such improvements. Exploiting FPGA-based platforms such as COPACOBANA [16] would also be an interesting direction of research. Note that although statistical attacks are well suited for FPGA implementations, other computing platforms could lead to similar speedups. As discussed, e.g. in [2] for the case of elliptic curves, cryptanalysis applications generally benefit from hybrid infrastructures (e.g. based on FPGAs, but also CPUs, GPUs, ASICs, . . .). As far as PRESENT is concerned, optimized implementations on these devices and cost comparisons with the FPGA design we propose in this paper would be another interesting scope for further research.

[3] The experiments presented in [6] reached a data complexity of $2^{35.6}$.

Next, our experiments confirm the previous theoretical predictions for statistical saturation attacks in [6, 7], for up to 18 rounds (and 19 rounds if a two-round partial decryption process was considered). They also provide hints that a statistical hull effect is appearing after 18 rounds of PRESENT. Confirming this effect with more confidence would require analyzing a few more rounds and was not possible within our current computational limits. In particular, extending our experiments for 18 and 19 rounds, and complexities up to 2^{51}, would be interesting. Nevertheless, the evaluations in this paper suggest that assumptions required to theoretically estimate the data complexity of statistical saturation attacks may not be respected beyond 24 rounds. Since the statistical hull effect we consider in this paper is close to the linear hull effect considered in linear cryptanalysis, one should probably question the validity of statistical attacks targeting more than 24-round PRESENT in general. Note that this question also holds for differential cryptanalysis, although the combination of several characteristics always increases the differential probability, because of the key dependencies implied by such a combination.

Finally, in most current block ciphers, the number of rounds needed to resist statistical cryptanalyses is determined based on Knudsen's practical security paradigm. But a more accurate technique would be to determine exactly when the statistical hull effects start to be effective in a cipher. In general, solving this problem is highly computationally intensive. The results in this paper lead to the interesting question whether the hull effect could be detected by sampling less than the full plaintext/key space. In case of a positive answer, a very interesting scope for further research would be to quantify this observation with robust statistics, in order to derive a new criteria for selecting the number of rounds in block ciphers. Analyzing small-scale block ciphers that can be exhaustively evaluated against different attacks could be a first useful step in this direction.

Acknowledgements. Stéphanie Kerckhof is a PhD student funded by a FRIA grant, Belgium. François-Xavier Standaert is a research associate of the Belgian fund for scientific research (FNRS-F.R.S). Work funded in part by the Belgian State's IAP program P6/26 BCRYPT.

References

1. Martin Albrecht and Carlos Cid. Algebraic techniques in differential cryptanalysis. In Dunkelman [9], pages 193–208.
2. Daniel V. Bailey, Lejla Batina, Daniel J. Bernstein, Peter Birkner, Joppe W. Bos, Hsieh-Chung Chen, Chen-Mou Cheng, Gauthier van Damme, Giacomo de Meulenaer, Luis Julian Dominguez Perez, Junfeng Fan, Tim Güneysu, Frank Gurkaynak, Thorsten Kleinjung, Tanja Lange, Nele Mentens, Ruben Niederhagen, Christof Paar, Francesco Regazzoni, Peter Schwabe, Leif Uhsadel, Anthony Van Herrewege, and Bo-Yin Yang. Breaking ecc2k-130. Cryptology ePrint Archive, Report 2009/541, 2009. `http://eprint.iacr.org/`.
3. Céline Blondeau and Benoît Gérard. Links between theoretical and effective differential probabilities: Experiments on present. Cryptology ePrint Archive, Report 2010/261, 2010. `http://eprint.iacr.org/`.

4. Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and C. Vikkelsoe. Present: An ultra-lightweight block cipher. In Pascal Paillier and Ingrid Verbauwhede, editors, *CHES*, volume 4727 of *Lecture Notes in Computer Science*, pages 450–466. Springer, 2007.
5. Joo Yeon Cho. Linear cryptanalysis of reduced-round present. In Josef Pieprzyk, editor, *CT-RSA*, volume 5985 of *Lecture Notes in Computer Science*, pages 302–317. Springer, 2010.
6. Baudoin Collard and François-Xavier Standaert. A statistical saturation attack against the block cipher present. In Marc Fischlin, editor, *CT-RSA*, volume 5473 of *Lecture Notes in Computer Science*, pages 195–210. Springer, 2009.
7. Baudoin Collard and François-Xavier Standaert. Multi-trail statistical saturation attacks. In Jianying Zhou and Moti Yung, editors, *ACNS*, volume 6123 of *Lecture Notes in Computer Science*, pages 123–138, 2010.
8. Baudoin Collard and François-Xavier Standaert. Experimenting linear cryptanalysis. to appear in Advanced Linear Cryptanalysis (book chapter), IOS Press, 2011.
9. Orr Dunkelman, editor. *Fast Software Encryption, 16th International Workshop, FSE 2009, Leuven, Belgium, February 22-25, 2009, Revised Selected Papers*, volume 5665 of *Lecture Notes in Computer Science*. Springer, 2009.
10. Benoît Gérard and Jean-Pierre Tillich. On linear cryptanalysis with many linear approximations. In Matthew G. Parker, editor, *IMA Int. Conf.*, volume 5921 of *Lecture Notes in Computer Science*, pages 112–132. Springer, 2009.
11. Miia Hermelin, Joo Yeon Cho, and Kaisa Nyberg. Multidimensional extension of matsui's algorithm 2. In Dunkelman [9], pages 209–227.
12. Liam Keliher, Henk Meijer, and Stafford E. Tavares. Improving the upper bound on the maximum average linear hull probability for rijndael. In Serge Vaudenay and Amr M. Youssef, editors, *Selected Areas in Cryptography*, volume 2259 of *Lecture Notes in Computer Science*, pages 112–128. Springer, 2001.
13. Liam Keliher, Henk Meijer, and Stafford E. Tavares. New method for upper bounding the maximum average linear hull probability for spns. In Birgit Pfitzmann, editor, *EUROCRYPT*, volume 2045 of *Lecture Notes in Computer Science*, pages 420–436. Springer, 2001.
14. Lars R. Knudsen. Practically secure feistel cyphers. In Ross J. Anderson, editor, *FSE*, volume 809 of *Lecture Notes in Computer Science*, pages 211–221. Springer, 1993.
15. Manoj Kumar, Pratibha Yadav, and Meena Kumari. Flaws in differential cryptanalysis of reduced round present. Cryptology ePrint Archive, Report 2010/407, 2010. `http://eprint.iacr.org/`.
16. Sandeep Kumar, Christof Paar, Jan Pelzl, Gerd Pfeiffer, and Manfred Schimmler. Breaking ciphers with copacobana - a cost-optimized parallel code breaker. In Louis Goubin and Mitsuru Matsui, editors, *CHES*, volume 4249 of *Lecture Notes in Computer Science*, pages 101–118. Springer, 2006.
17. Gregor Leander. On linear hulls, statistical saturation attacks, present and a cryptanalysis of puffin. to appear in the proceedings of Eurocrypt 2011.
18. Mitsuru Matsui. Linear cryptoanalysis method for des cipher. In *EUROCRYPT*, pages 386–397, 1993.
19. Jorge Nakahara, Pouyan Sepehrdad, Bingsheng Zhang, and Meiqin Wang. Linear (hull) and algebraic cryptanalysis of the block cipher present. In Juan A. Garay, Atsuko Miyaji, and Akira Otsuka, editors, *CANS*, volume 5888 of *Lecture Notes in Computer Science*, pages 58–75. Springer, 2009.

20. Kaisa Nyberg. Linear approximation of block ciphers. In *EUROCRYPT*, pages 439–444, 1994.
21. Kenji Ohkuma. Weak keys of reduced-round present for linear cryptanalysis. In Michael J. Jacobson Jr., Vincent Rijmen, and Reihaneh Safavi-Naini, editors, *Selected Areas in Cryptography*, volume 5867 of *Lecture Notes in Computer Science*, pages 249–265. Springer, 2009.
22. Onur Özen, Kerem Varici, Cihangir Tezcan, and Çelebi Kocair. Lightweight block ciphers revisited: Cryptanalysis of reduced round present and hight. In Colin Boyd and Juan Manuel González Nieto, editors, *ACISP*, volume 5594 of *Lecture Notes in Computer Science*, pages 90–107. Springer, 2009.
23. Meiqin Wang. Differential cryptanalysis of reduced-round present. In Serge Vaudenay, editor, *AFRICACRYPT*, volume 5023 of *Lecture Notes in Computer Science*, pages 40–49. Springer, 2008.
24. Xilinx. *Virtex-5 FPGA User Guide*, 2010. `http://www.xilinx.com/support/documentation/user_guides/ug190.pdf`.
25. Muhammad Reza Z'aba, Håvard Raddum, Matthew Henricksen, and Ed Dawson. Bit-pattern based integral attack. In Kaisa Nyberg, editor, *FSE*, volume 5086 of *Lecture Notes in Computer Science*, pages 363–381. Springer, 2008.

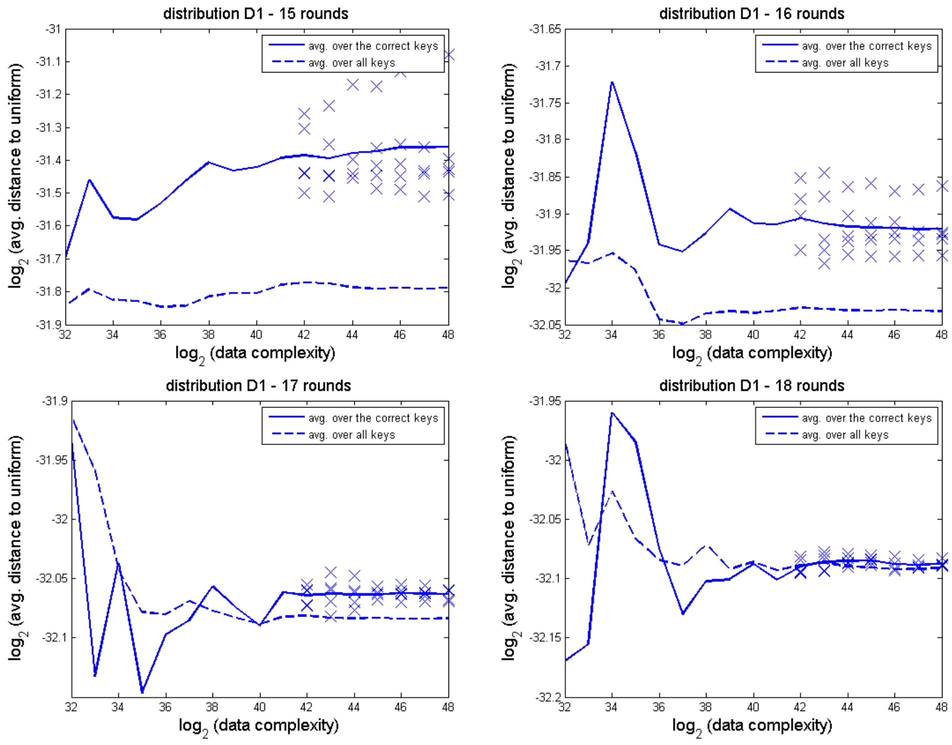


Fig. 10. Distance to uniform of distribution D1.

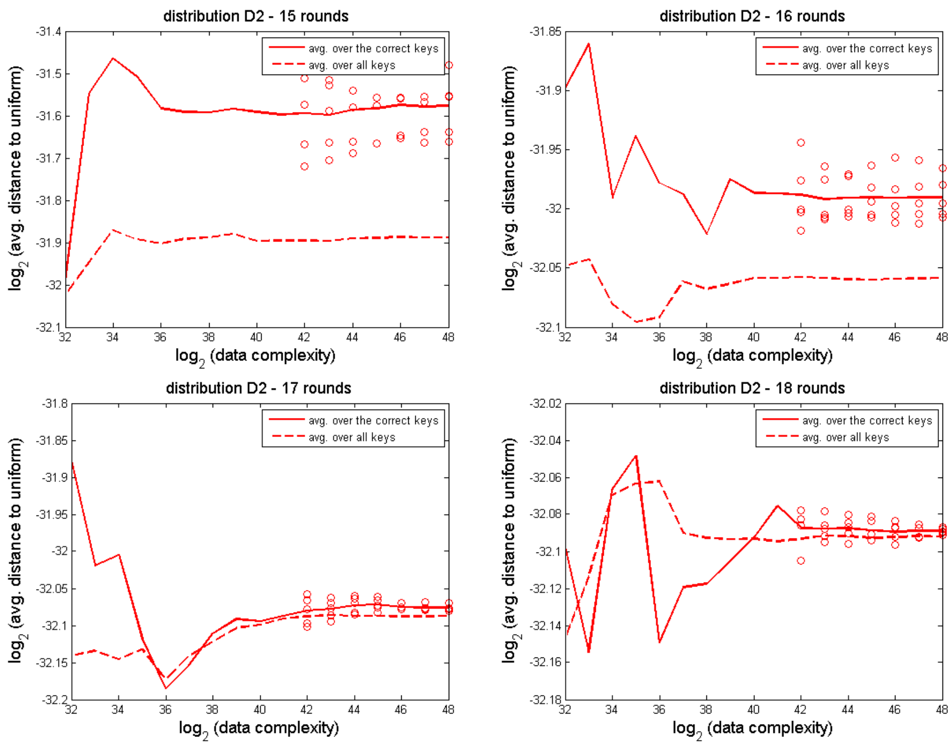


Fig. 11. Distance to uniform of distribution D2.