On the Synthesis of Attack Tolerant Cryptographic Hardware

J. Mathew[1], S. Banerjee[1], D.K. Pradhan[1], S.P. Mohanty[2], A.M. Jabir[3]
[1]Department of Computer Science, University of Bristol, UK.
[2]University of North Texas, Denton,TX 76203.
[3]Dept. of Comp. Sci & Electronics, Oxford Brookes University, UK.

***Abstract*— Concurrent error detection and correction is an effective way to mitigate fault attacks in cryptographic hardware. Recent work on differential power analysis shows that even mathematically-secure cryptographic protocols may be vulnerable at the physical implementation level. By measuring energy consumed by a working digital circuit, it is possible to gain valuable information about the encryption algorithms used and even the specific encryption keys. Thwarting such attacks requires a new approach to logic and physical designs. This paper presents a systematic approach to fault tolerant cryptographic hardware designs. Firstly, the effectiveness of the Hamming code based error correction schemes as a fault tolerance method in stream ciphers is investigated. Coding is applied to Linear Feedback Shift Registers (LFSR) based stream cipher implementations. The method was implemented on industrial standard stream ciphers, e.g. A5/1(GSM), E0 (Bluetooth), RC4 (WEP), and W7. The performance variation of stream cipher algorithms with error detection and correction was studied by synthesising the designs on Field Programmable Logic Arrays (FPGA) and Application Specific Integrated Circuits (ASIC). Further, we analyse hardware building blocks to minimise switching activity of a circuit over all possible inputs and input transitions by adding redundant gates and increasing the overall number of signal transitions. We also discuss the overhead and compositional properties of uniformly-switching circuits.**

I. Introduction

There is a growing demand for tighter security at banks and ATMs, airports, industrial sites, military installations, large entertainment complexes, and power stations. Designing hardware for such systems is particularly challenging because it must withstand tampering, fault attacks, and eavesdropping. It has recently been shown that for many digital signature and identification schemes an attacker can inject faults into the hardware and the resulting incorrect outputs may completely expose the secret signatures [2]. Hence, as cryptographic hardware becomes increasingly vulnerable to such fault attacks, future cryptographic processors must continue processing computations correctly in the presence of such attacks. In this regard, [3] presents a technique showing how the presence of faults in the public parameters of an elliptic curve cryptosystem may expose the secret keys.

Electronic cryptographic hardware can process information at significantly higher rates than mechanical devices, and can inadvertently leak much more information through side channels. A resourceful attacker may even be able to observe the performance of an electronic device on inputs of their choice, e.g., trying many passwords per second and watching out for an unusual activity pattern on a chip. Physical contact with the chip is not necessary, as demonstrated recently by electromagnetic-emission [12] and acoustic attacks [17] against public-domain implementations of RSA running on common laptops and desktops. Diagnostic equipment used by chip manufacturers for silicon debugging [15] can also be used to undermine cryptographically secure hardware. Thermal imaging with even poor resolution can distinguish inputs that cause unusual patterns and identify large structural components of the chip, such as multipliers, that are indispensable in public-key cryptography. This approach facilitates timing attacks where the attacker may glean useful information, e.g. during a multiply operation. Such attacks have been detailed by Kocher [13] and demonstrated by Boneh and Brumley in a practical setting against the OpenSSL [7]. In the meantime, more sophisticated spectrographic equipment is available and is rapidly becoming inexpensive. Additionally, attackers may compromise hardware systems through social engineering and gain unauthorised physical access to critical keyboards, monitors, LEDs or wires. Therefore, security analysis often ascribes seemingly unreasonable powers to the attackers, in the hope to err on the side of overestimating attackers rather than underestimating them. Many types of secure hardware (wireless base-stations, aerospace communications, e-cash) must last for over 10 years, and thus designers must account for future technological advances rather than only consider known attacks. This suggests frugal minimisation of side-channel information, new mathematical formalisations, and new design algorithms.

The proposal is to complicate attacks against cryptographically secure hardware that are based on Fault attacks and Differential Power Analysis (DPA) [14]. This statistical technique uses measurements of power consumption of a circuit in different circumstances in the hope that energy patterns may correlate with signal patterns. DPA-based methods automatically formulate numerous hypotheses about cryptographic patterns, e.g. digital keys, and correlate them against empirical side-channel information. While typical DPA attacks rely on measuring supply current, statistically significant data about thermal patterns may also weaken cryptographic hardware. Simply increasing the number of sensors, the frequency of readings, or the sensitivity of thermal measurements may automatically decrease expected discovery time for encryption keys.

In light of this, firstly, this paper proposes a fault tolerant

design of LFSR/GLFSR. We consider multiple error detection, in standard stream cipher implementation by sub dividing long blocks in small blocks and applying multiple hamming codes. The overhead for the proposed error correction technique is also analysed. We also study uniformly-switching versions of common logic blocks, such as adders and comparators, and cryptographic hardware-specific blocks such as Galois field multipliers. Given the overhead of uniformly-switching logic, we do not expect that complete secure systems will be entirely based on such circuits. For example, some cryptographic algorithms use published look-up tables that do not compromise secure information. Such look-up tables do not necessarily need to be protected from information leakage.

The remainder of the paper is organised as follows. Section II presents the preliminaries on stream ciphers and outlines the approach proposed in this paper, along with definitions considered in the rest of the paper. We construct uniformly-switching variants of various building blocks. In Section III we investigate error correction in Linear Feedback Shift registers. Section IV outlines various experimental results, and contrasts our approach with the others. Section V presents the conclusions.

II. Preliminaries

Linear Feedback Shift Registers (LFSR) are used in key-stream and Built-in Self Test (BIST) generators due to their simple hardware structures. They can produce sequences of large period, with good statistical properties. An LFSR of length N consists of N elements capable of storing one bit each. The Galois Linear Feedback Shift Register (GLFSR) is a solution for improving randomness of the pattern generators (PG). Besides generating random numbers, the GLFSR has fault tolerant properties [5]. The GLFSR is, as the name suggests, a generalised LFSR, which is defined over Galois Field GF(2^{∂}), $\partial > 1$. The GLFSR has three components: adders, multiplier, and memory elements. The feedback polynomial of an n-stage GLFSR can be represented as in Eq. 1.

$$Phi(y) = y^n + \Phi_{n-1}y^{n-1} + \cdots + \Phi_1 y + \Phi_0 \quad (1)$$

A general representation of GLFSR is shown in Fig. 1.

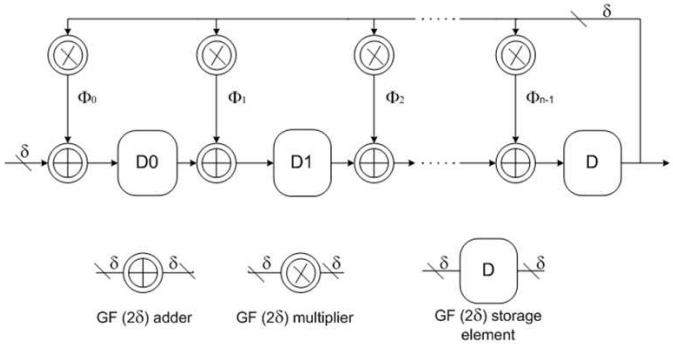

Fig. 1. Galois LFSR.

There are several stream cipher algorithms. In a binary additive stream cipher, the cipher text is produced by bit-wise addition of the plain text with a key-stream—all in binary. The key-stream generator is initialised using a secret key. The most common key-stream generator is a combination of several GLFSR/LFSRs that are combined together through a non-linear Boolean function. In this paper, we consider four standard stream ciphers and implement two versions (fault tolerant and non-fault tolerant) for comparing their performances. The A5/1, E0, RC4, and W7 are well-known stream ciphers, and they have been specified in popular standards and protocols. The A5/1 is a stream cipher used for encrypting over the air transmissions in the GSM standard [22]. A GSM conversation is transmitted as a sequence of 228-bit frames (114 bits in each direction) every 4.6 milliseconds. Each frame is XORed with a 228-bit sequence produced by the A5/1 key-stream generator. The initial state of this generator depends on a 64-bit secret key, K_c, which is fixed during the call duration, and on a 22-bit frame number. The encryption of packet payloads in Bluetooth is performed by the E0 stream cipher [24]. The W7 algorithm is a symmetric key stream-cipher that supports key lengths of 128 bits. The W7 cipher contains eight similar models, C1, C2,...,C8. Each model consists of three LFSRs and one majority function. The RC4 is a variable key-size stream cipher developed by Ron Rivest for the RSA Data Security, Inc. The RC4 stream cipher has two phases: the key set-up, and the key-stream generation. Both phases must be performed for every new key. During an n-bit key set-up the encryption key is used to generate an encrypting variable using two arrays (the state and the key array) and n-number of mixing operations. A detailed architecture can be found in [24].

One of the counter measures to mitigate DPA is to equalize power dissipation. When selecting gate libraries, we will require that every gate has the same number of switching outputs for every possible single-output transition. Therefore, when a multiple-input transition is decomposed into a shortest sequence of single-bit transitions, the overall result (in terms of power) does not depend on the specific sequence chosen. Formally, we define uniformly-switching gates as follows [4].

Definition 1: A gate (function) $f(x_1,x_2,...,x_n) = (y_1,y_2,...,y_k)$ with n inputs and k outputs is uniformly-switching iff there is a constant $0 < M_f \leq k$ with the following property: For any input combination $(x_1,x_2,...,x_n)$, changing the value of any single bit in it will lead to changing of exactly M_f output bits.

Definition 2: A circuit is called weak uniformly-switching if for each input wire x_i there exists a constant d_i such that for any one-bit input transition on x_i, the circuit experiences d_i gate switches. The vector $(d_1,d_2,...,d_n)$ is called the characteristic vector of the weak uniformly-switching circuit.

Definition 3: A circuit is called strong uniformly-switching with a fixed parameter C if for any one-bit input transition, it experiences C gate switches.

III. Error Correction

In this section, we present the proposed design of the error correctable LFSR. The basic principle is explained using a 4 bit example. The basic structure of a 4 bit block is shown in Fig. 2. The classical LFSR based on a typical primitive

polynomial is first designed. We computer check bits for each cycle for error detection and correction, which is based on the Hamming's principles. The Hamming codes are the simplest of a group of codes known as the linear block codes [18]. We divide large chunk of sequential elements in small blocks and encode the separately. Next, we present the algorithm for designing the proposed scheme with an example.

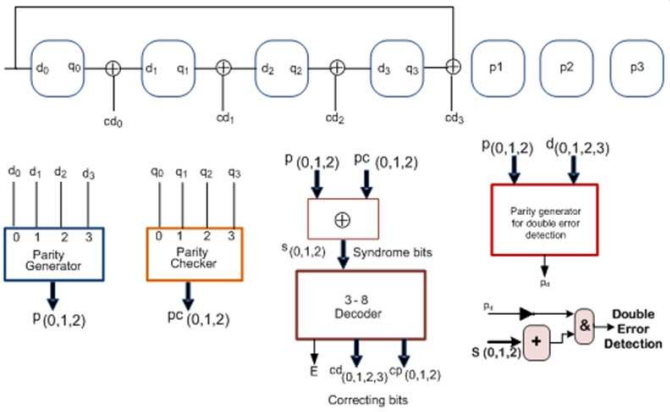

Fig. 2. Error correction in a 4 bit block.

Let $\vec{d} = [d_0, d_1, d_2, \ldots, d_{n-1}]$ be the input of the seqential elements in an LFSR and $\vec{q} = [q_0, q_1, q_2, \ldots, q_{n-1}]$ the output. Also let r be the number of parity bits, and $\vec{p} = [p_0, p_1, \ldots, p_{r-1}]^T$ and $\vec{pc} = [pc_0, pc_1, \ldots, p_{r-1}]$ respectively be the predicted and the parity bits generated from the output bits. Let **H** be the parity check matrix associated with the proposed single error correction scheme.

Design Procedure:

- Determine the number of block checkers (ms) required to satisfy the equation $m = \lfloor n/k \rfloor$, where k is number of error corrections.
- Determine the number of parity bits (r) required to satisfy the equation $m + r + 1 \leq 2^r$.
- Construct the **H** matrix, with $(m + r)$ non-zero r-bit column vectors. The dimension of the resulting matrix is $r \times (m+r)$.
- A column vector with a single 1 is assigned to parity P_i.
- The column vector with all 1s is assigned to output bit c_{m-1}.
- The remaining m columns are assigned the output bits c_i, without any constraints.
- Generate the parity expressions in terms of d_is.
- For DED per block, choose the parity check matrix such that the output bits are assigned to the columns with odd number of ones. In this case additional parity bits maybe required.
- Finally, combine the block register, check bit register, output encoder, decoder, and the correction logic as shown in 2.

The following example illustrates the above design procedure.

Example 1: Consider a four bit LFSR structure over constructed in Fig. 2. Here we have $m = 4$. Therefore, we need 3 parity bits to correct single errors. We have,

$$\mathbf{H} = \begin{array}{c} \begin{matrix} p_0 & p_1 & p_2 & d_0 & d_1 & d_2 & d_3 \end{matrix} \\ \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix} \end{array}.$$

Therefore, the parity check equations are: $p_0 = d_0 + d_1 + d_3; p_1 = d_0 + d_2 + d_3; p_2 = d_1 + d_2 + d_3$.

Error Correction: The error correction procedure involves three steps: (i) compute the output parities, (ii) computer syndrom (compare with stored parity bits), and (iii) decode the syndrome and apply corrections. Let us analyse the various error scenarios on the proposed architecture and their effects. First, a malicious attack/error on one of the registers is considered. Consider Fig. 2 with the first register bit in error. Here, we refer to Example 1 with the single error correcting **H** matrix. Let the input of the LFSR be $d = (d_0, d_1, d_2, d_3) = (1,0,0,0)$. The check bits stored are $p = (p_0, p_1, p_2) = (110)$. Let us assume that , an error in the first bit causes an erroneous output $q = (q_0, q_1, q_2, q_3) = (0,0,0,0)$. At the output the output parity bits $pc = (pc_0, pc_1, pc_2) = (0,0,0)$. Comparing the output parity bits with the stored parity bits gives the syndrome $(1,1,0)$ and the bit corresponding to this syndrome is d_0 (see the **H** matrix). Therefore, the first bit gets automatically corrected as shown. In Fig. 2 (cd_0, cd_1, cd_2, *and* cd_3 are the correcting signals. In this example (cd_0 set high by the decoding logic to correct d_0. Second, a malicious attack/error on the parity check bit is considered. In this case an attack/error on the parity would not cause an error in the functional output and the syndrome generated will be one of the following: $\{(0,0,1),(0,1,0),(1,0,0)\}$. Since these syndromes are not decoded, no correction is applied. With a similar argument, any error in the functional registers that causes single bit error in the output can be corrected by the above technique, whereas for the errors in the parity circuit that causes single error in the predicted parities are not corrected. Therefore, by introducing the parity check bits will not compromise reliability.

A. Uniformly-Switching Gates and Circuits

For synthesis of uniformly-switching circuits we are going to use a complete gate library consisting of uniformly-switching gates [4]. One-input one-output buffer and inverter gates are uniformly-switching, but we also need more powerful gates. Adding two-input one-output uniformly-switching gates is not sufficient either. Such a gate library only allows to compute all Boolean functions that are with respect to the bitwise XOR operation. In general, any negation and/or permutation of the outputs of a uniformly-switching function produces another uniformly-switching function. Detailed analysis of uniformly-switching variants of blocks are discussed in section IV.

IV. Experimental Results

This section presents the hardware and performance analysis of the proposed technique, which reflects its usefulness. For this purpose, the original algorithm of the four stream ciphers A5/1, E0, W7, and RC4 were implemented in VHDL. The designs were simulated in Modelsim to verify the functionality. Further, the hardware variation with the usage of different

optimal register blocks for parity generation is investigated. For example, Fig. 3 shows the possible different groupings for an 128-bit LFSR. The degrees of the LFSRs in each cipher were varied from 64 to 128. The GLFSRs were considered as registers and divided into groups of optimal register sets. The parity bits required in each case and the errors corrected were compared with the area consumed. Fig. 4 shows the comparison with various combination of parity check bits.

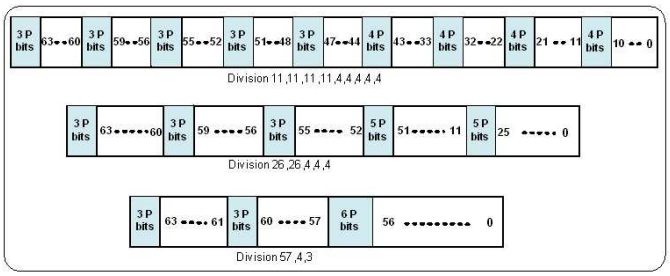

Fig. 3. Register grouping for multiple Hamming codes.

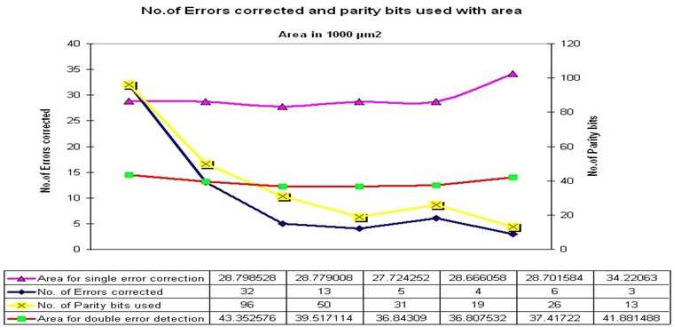

Fig. 4. Comparison of multiple error correction.

The technique most related to ours includes the *SABL* logic family [19] and the recent *Wave Dynamic Differential Logic (WDDL)* [21], both from UCLA. A major advantage of WDDL is that it can be handled by a traditional EDA tool flow. Our approach goes further in the sense that we show how to reuse existing tools for synthesis and layout. However, we pursue a somewhat different task—equalising energy dissipation, and not total power consumption. Indeed, in CMOS power is consumed mostly during the 0-1 transitions, but both the 0-1 and 1-0 transitions dissipate energy.

We observe that empirical results in [21] require careful interpretation. For example, the path delay overhead in Table 1 does not account for the use of every second cycle in the WDDL circuits for "pre-charging waves", which halves data rate for the same cycle time.

Our techniques affect data rate due to the gate sizing, and, as experiments in Section IV-A show, the overhead is marginal. Area and delay overhead may strongly depend on the logic functions. Indeed, the WDDL logic requires re-expressing each Boolean function using AND/OR/NOT gates, while the uniformly-switching logic extends existing circuits. The former results in a large overhead when many XOR operations are required, e.g. in the Kasumi algorithm [21]. Our techniques adapt existing circuits and do not impose significant restrictions on the gates used. They incur the smallest overhead on circuits consisting entirely of the XOR, XNOR, and NOT gates. In particular, [20] reported a 3 times ($\approx \frac{2.45}{0.79}$) area increase for the implementation of AES, while using our construction the area increase is at most 2 times (in the worst case, using the doubling construction on top of the AES implementation which is linear). In addition, in our algorithm the data rate is not slowed down by the design to such extend as it is for WDDL.

Since in WDDL all the inputs are pre-charged to zero, the use of AND/OR gates instead of NAND/NOR seems unavoidable, implying additional 50% area overhead in CMOS (however, a convenient LUT-based implementation is proposed in [21]). This and the pairing of AND/OR gates in WDDL implies a lower bound of 3× on the area overhead, which agrees with the empirical data in [21, Table 1] and sharpens the lower bound of 2× as presented in [21]. The use of WDDL seems to require complete re-synthesis, whereas we adapt existing circuits and preserve the structure of critical paths.

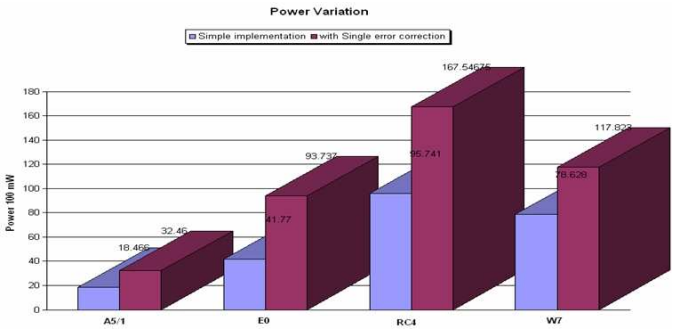

Fig. 5. Power Comsumption: original vs. fault tolerant

A. Area, Timing and Power Analysis

The two versions (classical and weak uniformly-switching) of typical combinational logic blocks have been designed and coded in VHDL. The test circuits were divided into two groups: (1) AND/OR functions and basic arithmetic circuits, and (2) AND-XOR intensive, such as Galois multiplier circuits. This was done to validate our basic constructs and demonstrate that our technique extends to larger circuits. For comparison, the doubling contruction version of the Galois multiplier circuits were also designed. The arithmetic circuits designed over the Galois fields are crucial to the implementation of certain cryptographic algorithms, such as the elliptic curve cryptography. The designs were simulated using ModelSim^{TM} and were tested for functionality by giving various inputs. The designs were synthesized using the Synopsys tools in the UMC technology library, using the 0.18 micron CMOS technology. The Synopsys Power Compiler^{TM} was used to estimate the power consumptions. The area, delay, and power estimates for the basic circuits are shown in Table II. As expected, the overhead varies depending on the type of circuit and logic elements used. For instance, AND/OR dominated circuits are expensive, requiring up to 144% extra area to make them weak uniformly-switching. As mentioned earlier, since

the proposed design is for cryptographic applications, we have also analysed various Galois field circuits. For Galois multipliers, which are widely used in cryptographic processor designs, there is approximately 90% area and power overhead to make them weak uniformly-switching. The area, delay, and power analysis for the various Galois field multiplier designed with the various approaches (original, weak uniformly switching, and using the doubling construction) are shown in Figs. 6, 7, and 8 respectively. As explained in the design flow section, first the designs were made weak uniformly switching and then the doubling construction was applied. The actual values are normalised with respect to the original design. The overhead is on an average about 300% after doubling construction. There is a slight delay overhead in transition from the original design to the doubled variant due to the extra XOR gates used in the doubling construction, whereas there is no delay difference between the original and weak uniformly switching versions. Since the analysis presented here is based on standard-cell CMOS layouts, we have focused mostly on validating the key concepts proposed and estimating the power, area, and delay overhead. However, further optimisations are possible at transistor level.

TABLE I

SYNTHESIS RESULTS OF VARIOUS DESIGNS

Prim Design	ASIC (*area in* um^2)		FPGA (LUTs)	
	Original	with EC	Original	with EC
A5/1	33420.3	46558.2	365	445
E0	70315.0	111354.3	607	936
RC4	244687.1	401162.1	3627	5803
W7	36924.5	59505.5	373	622

TABLE II

UNIFORMLY-SWITCHING VARIANTS OF VARIOUS CIRCUITS.

circuit	*area* (um^2)	*delay* (ns)	*power*(uw) (uw)	*area* (um^2)	*delay* (ns)	*power* (uw)
Full adder	90.1	0.43	13.5	197.1	0.43	29.4
Mux-2-1	51.6	0.28	5.6	139.90	0.28	24.4
Decoder-2-4	64.5	0.16	9.1	116.9	0.16	16.8
Multiplier (2bit)	132.6	0.45	13.6	222.6	0.45	31.76
4bit adder/ subtractor	787.04	1.34	140	1593	1.34	226.3

B. Spice Simulation Results

Both the proposed architecture and the classical designs have been implemented in CMOS transistor level using the HSPICE tool in the 0.18 micron technology. Fig. 9 shows the simulated power trace and its histogram of a GF(64) multiplier circuit. The designs have been simulated with a 20 ns clock period for all the 4096 input transitions. Fig. 10 shows the weakly switched variant of Fig. 9. It should be noted that the proposed design has more uniform power consumption irrespective of the switching activity. We also considered 10% process variation in the design parameters. The process variation study conducted relies on the well known Monte Carlo method that could be done in a reasonable amount of time. Fig. 11 shows the power consumptions for 100 Monte Carlo simulations. We noticed more or less random changes in the power consumptions as we varied the process parameters.

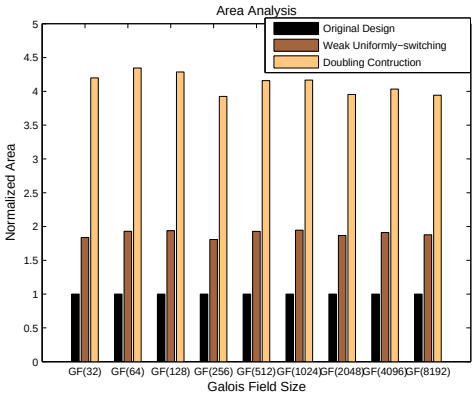

Fig. 6. Area analysis of various Galois field multipliers and their uniformly-switching variants.

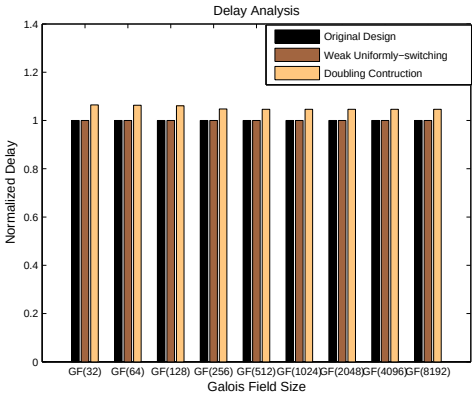

Fig. 7. Delay analysis of various Galois field multipliers and their uniformly-switching variants.

V. CONCLUSIONS

In this paper we proposed a systematic design approach for fault tolerant cryptography hardware. Firstly, we analysed the effectiveness of Hamming code based fault tolerance in stream ciphers. The method was implemented on industrial standard stream ciphers, e.g. A5/1(GSM), E0 (Bluetooth), RC4 (WEP), and W7. The performance variation of stream cipher algorithms with error detection and correction was studied by synthesising the designs on FPGAs and ASICs. Further, we explored the uniformly-switching logic for cryptographic applications to mitigate side-channel attacks based on dissipated energy. The cumulative area overhead is less than two times the area occupied without the redundancies. It is still affordable when DPA-resistance is a bottleneck but circuit size is not. The uniformly-switching logic incur particularly small overhead for deeply pipelined circuits and many arithmetic circuits. Further reducing the area and delay overhead of uniformly-switching logic is a major direction for future research. Additionally, ASICs may be bottlenecks for some low-power applications. Developing low-power cryptography is another research direction [1].

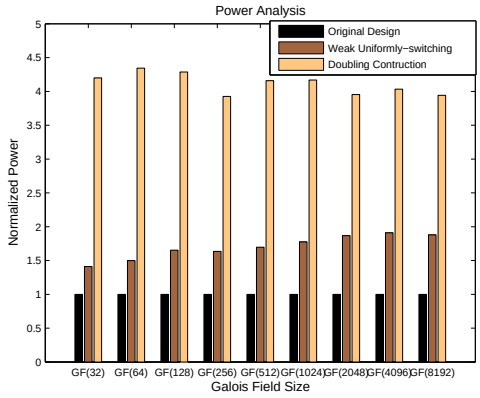

Fig. 8. Power analysis of various Galois field multipliers and their uniformly-switching variants.

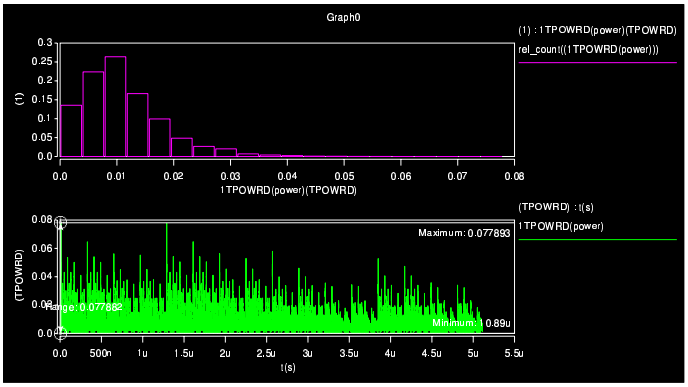

Fig. 9. Histogram and simulated power trace of classical designs.

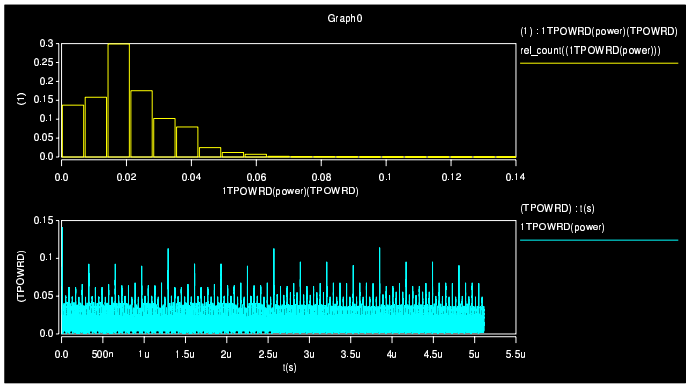

Fig. 10. Histogram and simulated power trace of weakly uniformly-switched variants.

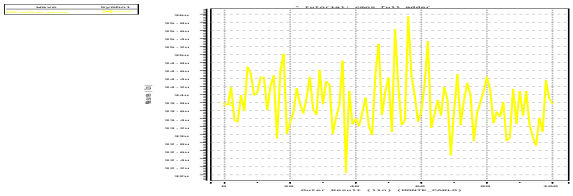

Fig. 11. Power consumptions under 10 percent process variation.

REFERENCES

[1] L. Benini, A. Macii, E. Macii, E. Omerbegovic, F. Pro, M. Poncino, "Energy-Aware Design Techniques for Differential Power Analysis Protection", *Design Automation Conference (DAC)*, pp. 36–41, June 2003.

[2] D. Boneh, R. A. DeMillo, and R. J. Lipton. On the Importance of Eliminating Errors in Cryptographic Computations. Journal of Cryptology, 2001.

[3] M. Ciet and M.J. Dueck. Elliptic Curve Cryptosystems in the Presence of Permanent and Transient Faults. Designs, Codes and Cryptography, July 2006.

[4] I. Markov and D. Maslov, "Uniformly-switching Logic for Cryptographic Hardware", *Design Automation and Test in Europe (DATE)*, pp. 432–433, March 2005.

[5] M. Chatterjee and D.K Pradhan "A GLFSR-Based Pattern Generator for Near-Perfect Fault Coverage in BIST", IEEE Transactions on Computers, Vol. 52, pp. 1535-1542, Dec. 2003.

[6] L. Benini, G. De Micheli, A. Macii, E. Macii, M. Poncino, R. Scarsi, "Glitch Power Minimization by Gate Freezing", *Design Automation and Test in Europe (DATE)*, pp. 163–167, 1999.

[7] D. Boneh and D. Brumley, "Remote Timing Attacks Are Practical", *12th Usenix Security Symposium*, 2003.

[8] J.-S. Coron, "Resistance Against Differential Power Analysis for Elliptic Curve Cryptosystems". *Workshop on Crypto Hardware and Embedded Systems* (CHES), pp. 292–302, 1999.

[9] J. Dj. Golic and C. Tymen, "Multiplicative Masking and Power Analysis of AES". *Workshop on Cryptographic Hardware and Embedded Systems* (CHES), pp. 198-212, 2002.

[10] International Biometric Group, "Biometrics Market and Industry Report 2004-2008", www.biometricgroup.com/reports/public/market_report.html, 2004.

[11] K. Itoh, J. Yajima, M. Takenaka, N. Torii, "DPA Countermeasures by Improving the Window Method", *Workshop on Cryptographic Hardware and Embedded Systems* (CHES), pp. 303–317, 2002.

[12] W. Knight, "Computer Chip Noise May Betray Code". *New Scientist*, May 2004. http://www.newscientist.com/news/news.jsp?id=ns99994979.

[13] P. Kocher, "Timing Attacks On Implementations Of Diffie-Hellman, RSA, DSS, and other systems", *Advances in Cryptology*, pp. 104–113, 1996.

[14] P. Kocher, J. Jaffe, and B. Jun, "Differential Power Analysis", *Lecture Notes in Computer Science*, 1666:388–397, January 1999.

[15] J. Markoff, "Intel Technicians Use Delicate Silicon Surgery to Fine-Tune Microchips", *New York Times*, 08/09/2004, p. C4.

[16] S. Morioka and A. Satoh, "An Optimized S-Box Circuit Architecture for Low Power AES Design", *Workshop on Cryptographic Hardware and Embedded Systems* (CHES), pp. 172–186, 2002.

[17] A. Shamir and E. Tromer, "Acoustic Cryptanalysis", http://www.wisdom.weizmann.ac.il/~tromer/acoustic/, 2004.

[18] W. Hamming. Error Detecting and Error Correcting Codes. Bell Systems Tech. Journal, pages 147160, 1950.

[19] K. Tiri, M. Akmal, I. Verbauwhede, "A Dynamic and Differential CMOS Logic With Signal-Independent Power Consumption to Withstand Differential Power Analysis on Smart Cards" *IEEE European Solid-State Circuits Conference (ESSCIRC)*, pp. 403-406, 2002.

[20] K. Tiri, D. Hwang, A. Hodjat, B. Lai, S. Yang, P. Schaumont, I. Verbauwhede, "A Side-Channel Leakage Free Coprocessor IC in 0.18m CMOS for Embedded AES-based Cryptographic and Biometric Processing", *Design Automation Conference (DAC)*, pp. 222–227, June 2005.

[21] K. Tiri and I. Verbauwhede, "A Logic Level Design Methodology for a Secure DPA Resistant ASIC or FPGA Implementation", *Design Automation and Test in Europe (DATE)*, pp. 246–251, 2004.

[22] "Recommendation GSM 02.09", European Telecommunications Standards Institute (ETSI), Security aspects".

[23] "Specification of the Bluetooth system", vol. 1.1, Feb. 2001. http://www.bluetooth.com/dev/specifications.asp.

[24] M. Galanis, P. Kitsos, G. Kostopoulos, N. Sklavos, and C. Goutis, "Comparison of the Hardware Implementation of Stream Ciphers" International Arab Journal of Information Technology, Vol. 2, No. 4, October 2005