Side-Channel Resistant Implementations of a Novel Lightweight Authenticated Cipher with Application to Hardware Security

Abubakr Abdulgadir, Sammy Lin, Farnoud Farahmand, Jens-Peter Kaps, Kris Gaj
{aabdulga,slin5,ffarahma,jkaps,kgaj}@gmu.edu
George Mason University
Fairfax, VA, USA

ABSTRACT

Lightweight authenticated ciphers are crucial in many resource-constrained applications, including hardware security. To protect Intellectual Property (IPs) from theft and reverse-engineering, multiple obfuscation methods have been developed. An essential component of such schemes is the need for secrecy and authenticity of the obfuscation keys. Such keys may need to be exchanged through the unprotected channels, and their recovery attempted using side-channel attacks. However, the use of the current AES-GCM standard to protected key exchange requires a substantial area and power overhead. NIST is currently coordinating a standardization process to select lightweight algorithms for resource-constrained applications. Although security against cryptanalysis is paramount, cost, performance, and resistance to side-channel attacks are among the most important selection criteria. Since the cost of protection against side-channel attacks is a function of the algorithm, quantifying this cost is necessary for estimating its cost and performance in real-world applications. In this work, we investigate side-channel resistant lightweight implementations of an authenticated cipher TinyJAMBU, one of ten finalists in the current NIST LWC standardization process. Our results demonstrate that these implementations achieve robust security against side-channel attacks while keeping the area and power consumption significantly lower than it is possible using the current standards.

CCS CONCEPTS

• **Security and privacy → Symmetric cryptography and hash functions**; **Hardware security implementation**.

KEYWORDS

lightweight cryptography; hardware; FPGA; side-channel attacks

ACM Reference Format:
Abubakr Abdulgadir, Sammy Lin, Farnoud Farahmand, Jens-Peter Kaps, Kris Gaj. 2021. Side-Channel Resistant Implementations of a Novel Lightweight Authenticated Cipher with Application to Hardware Security. In *Proceedings of the Great Lakes Symposium on VLSI 2021 (GLSVLSI '21), June 22–25, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3453688.3461761

GLSVLSI '21, June 22–25, 2021, Virtual Event, USA

ACM ISBN 978-1-4503-8393-6/21/06…$15.00
https://doi.org/10.1145/3453688.3461761

1 INTRODUCTION

To protect the intellectual property (IP) from being reverse engineered and misused by any participant of the manufacturing supply chain, various forms of hardware obfuscation have been developed. Obfuscation is the process of hiding the true functionality of an integrated circuit (IC). This functionality remains incorrect without the knowledge of a special key called an obfuscation key. The obfuscation key needs to be protected using cryptographic methods when this key is sent from the Authentication Server to an obfuscated IC during activation. The role of the Authentication Server can also be played by a trusted chip physically attached to an untrusted chip during packaging in the trusted facility. In [2], Communication and Obfuscation Management Architecture (COMA) has been proposed to handle secure and efficient distribution of obfuscation keys. The crucial component of this architecture is a hardware implementation of an authenticated cipher protected against side-channel attacks. This authenticated cipher must be trusted, but its implementation should introduce the smallest possible overhead in terms of area and power. In cryptography, trust comes from standardization. Until now, the only standardized authenticated cipher suitable for the protection of obfuscation keys is AES-GCM.

However, in 2018, as a response to the increasing need for lightweight cryptography (LWC) solutions, the U.S. National Institute of Standards and Technology (NIST) started an effort to select and standardize additional algorithms. In March 2021, NIST announced ten algorithms as finalists of this process. For such a selection process, security and software efficiency are usually the major criteria in the first rounds, with concentration shifting toward hardware and security against side-channel analysis (SCA) in the final rounds. One of the finalists that distinguished itself from others in terms of the minimal area, power, and energy usage is TinyJAMBU [13].

In this paper, we investigate the suitability and advantages of using TinyJAMBU instead of AES-GCM to protect obfuscation keys. As a part of this evaluation, we develop the first reported hardware implementation of TinyJAMBU protected against side-channel attacks (SCAs).

One of the most serious SCAs is Differential Power Analysis (DPA)[11]. In this attack, an adversary collects several power traces from the device performing a cryptographic function and uses statistical distinguishers to reveal sensitive information such as secret keys. First-order DPA exploits leakage that appears in the first-order statistical moment. On the other hand, high-order DPA uses leakage in high-order statistical moments. Implementations protected against *dth*-order DPA can be broken using *d+1*-order DPA. However, the number of leakage traces needed increases exponentially with respect to d [4].

The power and the practicality of DPA fueled a focus on countermeasures since the time the attack was published. Among existing countermeasures, masking style countermeasures, such as Threshold Implementations (TI) [15] and Domain Oriented Masking (DOM) [9], are provably secure under certain assumptions. In this work, we utilize the DOM scheme, which can be used to implement ciphers that are synthesizable for an arbitrary protection order. This is useful in providing flexible designs that adapt to the needs of applications with different levels of security and, at the same time, maintain a single code base.

In particular, we present a flexible SCA-protected hardware design of TinyJAMBU that can be synthesized for arbitrary order of protection utilizing the DOM countermeasure. The implementations can trade area for performance. We then show the advantages of this solution, as compared to the use of the existing standard AES-GCM, to protect obfuscation keys in COMA.

2 BACKGROUND

2.1 TinyJAMBU

TinyJAMBU [17] is a NIST LWC finalist designed to perform authenticated encryption. TinyJAMBU is a lightweight variant of JAMBU, a CAESAR Round-3 candidate that has a small footprint.

Figure 1 shows the data flow in TinyJAMBU. The TinyJAMBU-128 variant has a 128-bit state and uses a 128-bit key, 96-bit nonce, and 64-bit tag. The data (associated data and plaintext/ciphertext) is processed in 32-bit blocks. To update its state, TinyJAMBU uses a keyed permutation based on a Nonlinear Shift Register (NLFSR) as shown in Figure 2. To perform the permutation, the NLFSR is updated by calculating the *feedback* bit(s) as a function of the state and the key. Then the state register is shifted to the right. Depending on how the permutation is implemented, multiple feedback bits can be calculated in parallel.

2.2 Domain Oriented Masking

Hardware implementations of cryptography can leak information as a result of glitches[12]. The Domain-Oriented Masking (DOM) [10] countermeasure is designed to provide security in the presence of glitches, and, at the same time, implementations can be synthesized for arbitrary order of protection.

Similar to traditional Boolean masking, sensitive data is split into shares. For example x is split into x_0 and x_1, such that $x = x_0 \oplus x_1$. DOM, however, introduces the concept of share domains, where each share of a variable is associated with a domain. For example, x_0 and y_0 can be associated with *Domain0* and x_1 and y_1 can be associated with *Domain1*. The data in each domain is kept independent from other domains. If data from different domains must mix (e.g., in non-linear operations), precautions are taken to preserve the Independence. These precautions include adding randomness and using synchronization registers to stop glitches.

3 METHODOLOGY

We used the unprotected hardware implementation of TinyJAMBU at [3] as baseline to implement our SCA-resistant implementation. This design uses the RTL methodology which provides cycle-accurate operation simplifying side-channel protection. We employ the DOM countermeasure to build a TinyJAMBU implementation that can be synthesized for arbitrary order of protection only by setting a single configuration parameter d.

3.1 Unprotected Implementation

The lightweight hardware implementation of TinyJAMBU at [3] has a very small footprint requiring only 591 LUTs in Artix7 FPGA and at the same time provides a throughput of 250 Mbps. Our hypothesis, which is later confirmed by concrete results, is that we can implement high-order DPA resistant implementations of TinyJAMBU and still keep the implementation lightweight.

The baseline implementation is fully compatible with the LWC Hardware API. This enables direct comparison with other LWC candidates that adhere to the same API. Data communication with the outside world is performed in 32-bit words, which are equal to the block size, yielding an efficient design. The CipherCore is composed of a datapath, shown in Figure 3, which handles all computations, and a control Finite State Machine (FSM), which sequences the operations and controls data communication. The NLFSR is designed to be capable of computing between 1 and 32 feedback bits in one clock cycle. This value can be selected during synthesis using a parameter N. The implementation works as follows: The key is accepted 32-bits at a time and stored internally in a shift register. Afterward, the nonce is accepted, and the initialization phase is completed. The associated data is absorbed in 32-bits and used to update the state. On encryption, the plaintext is absorbed and, at the same time, the ciphertext is generated. Finally the tag is generated. When performing decryption, the ciphertext is absorbed and plaintext is produced. Finally, the tag is calculated and compared to the expected tag, and the status code is emitted based on the comparison outcome.

3.2 Protected Implementations

Building on the unprotected design described above, we developed a protected design that can be synthesized for arbitrary order of protection using the DOM countermeasure. Our protected design is parameterized by two synthesis parameters d and N, denoting the order of protection and the number of feedback bits computed in parallel, respectively. This highly flexible design with two degrees of freedom can be synthesized to cater to various applications with different security, cost, and performance requirements and at the same time maintained in a single code base.

Figure 4 depicts the datapath of a first-order protected implementation. The NLFSR_pr unit is the protected version of the NLFSR used for the permutation. Depending on the protection order, the state register is duplicated as needed and also the logic needed to compute the feedback bits. The rest of the datapath, which is responsible for implementing the mode of operation and performs data selection (multiplexing), FrameBits bit addition, etc., is denoted TinyJAMBU_ops in Figure 4. This component is duplicated $d + 1$ times, with each instance residing in a separate domain. Note that constants (FrameBits and partial block length) are added to the state only in Domain0 to avoid negating the operation.

To provide the random bits required for the DOM countermeasure, we utilize a Pseudo-Random Number Generator (PRNG) based on Trivium [6] which is seeded using data provided through the RDI port.

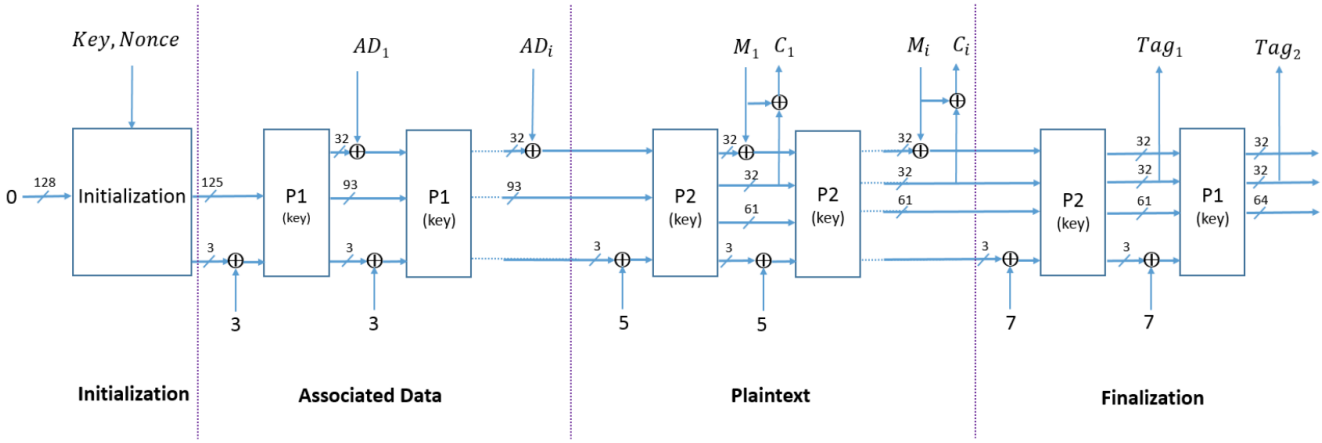

Figure 1: TinyJAMBU Authenticated Encryption [17]

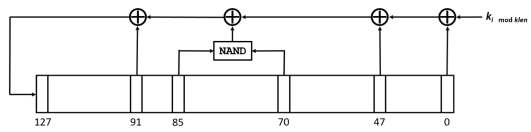

Figure 2: TinyJAMBU NLFSR [17]

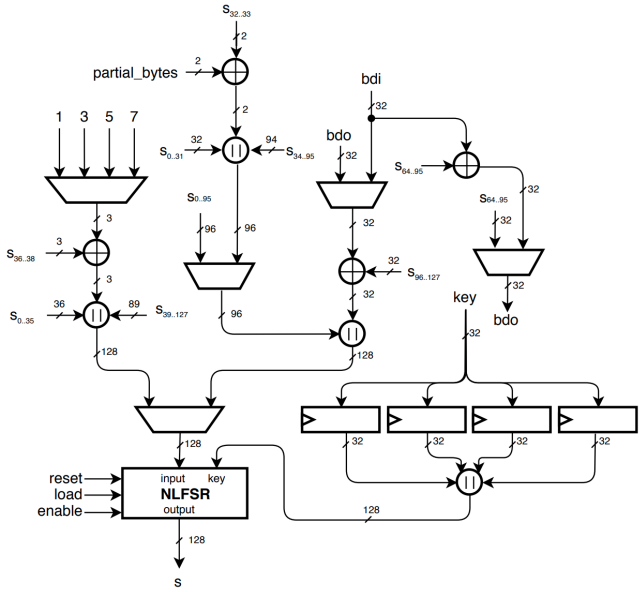

Figure 3: Baseline Unprotected TinyJAMBU Datapath

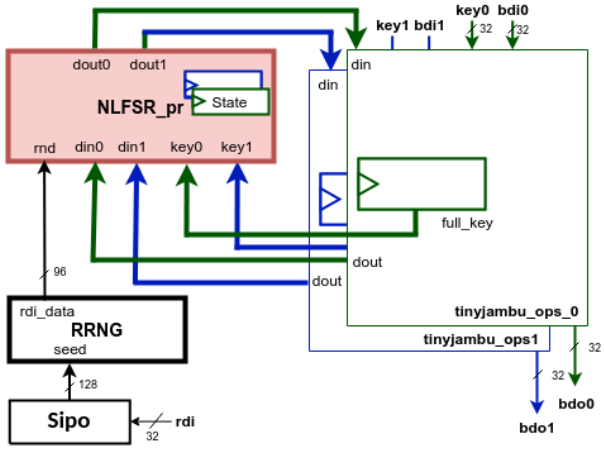

Figure 4: First-Order Protected TinyJAMBU Top-level Block Diagram

Figure 5 shows the block diagram of a second-order protected NLFSR. This component is critical since the only non-linear component in TinyJAMBU is DOM_NAND in the NLFSR. We instantiate a DOM-dep multiplier [9] that can be synthesized for arbitrary order instead of the AND gates in the unprotected design. We utilized the DOM-dep multiplier to avoid any leakage related to dependently shared variables. Since the multiplier is the only non-linear part of the design, it is the only component that is not residing purely in a separate share domain. To turn the AND gates to NAND, we negate Domain0 output of the each AND gate, which is equivalent to XOR with 1. Due to the mandatory synchronization registers in the DOM-dep multiplier, the NLFSR takes two clock cycles to compute N feedback bits. This will have the effect of doubling the number of clock cycles needed to perform the permutation. The rest of the NLFSR logic and the state are duplicated depending on the number of share domains, which is equal to $d + 1$.

The operation of the protected implementation is as follows: The state is cleared, and the seed is accepted from the RDI port and stored in a Serial-In Parallel-Out (SIPO) shift register. The PRNG is seeded and allowed to be initialized. Once the PRNG is ready, the input is accepted, and the operation proceeds similarly as in the unprotected design discussed above. The main exception is that

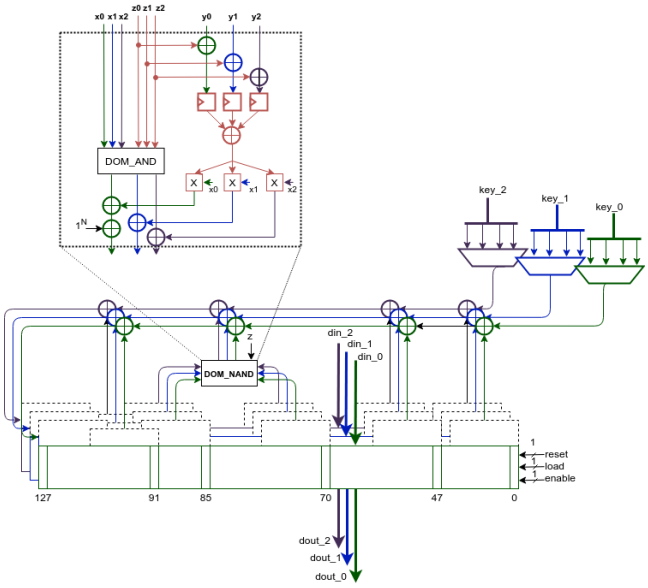

Figure 5: Protected Second-Order NLFSR. Thick lines are 128-bit buses. All other buses are N bit-wide unless specified directly. Data is processed in three share domains 0, 1 and 2.

data shares are processed in parallel by logic residing in separate domains.

Processing the shares in parallel means that the number of clock cycles to process a block remains the same as the protection order d increases, and the throughput may only be affected by minor variation in the maximum frequency of the implementation.

4 RESULTS

4.1 Countermeasure Verification

To validate the SCA resistance of our protected implementations, we utilized the Test-vector Leakage Assessment methodology (TVLA) [8] which is widely used to determine if a cryptographic device is leaking information without performing attacks. Techniques from [16] were utilized to perform high-order TVLA. We adopt a threshold of $| \ t \ | > 4.5$, which indicates that DUT leaks information at a confidence level of 99.999%. We utilized the Flexible Opensource workBench fOr Side-channel analysis (FOBOS) [1] platform, and the DUT was instantiated in a NewAE CW305 board, which is a low-noise SCA target board that uses Artix-7 (xc7a100tftg256-3) FPGA. The DUT power consumption is measured at the output of the CW305's Low-Noise Amplifier (LNA), that amplifies the voltage drop across the on-board 0.1 Ω shunt resistor. We clocked the DUT with a low frequency of 1.25 MHz to avoid signal distortion seen at high frequencies when power consumption at adjacent clock cycles interferes with each other. A USB3-based oscilloscope (Picoscope 5000) was used to collect traces at a sampling rate of 125 MS/s and 8-bit sample resolution.

TVLA results are shown in Figure 6. For the unprotected baseline design, the t values exceed the threshold, which indicates a significant leakage as expected. This leakage is observed even at only 10,000 traces. On the other hand, the TVLA for the first and second-order protected implementations show no observable leakage even when 1 million traces are analyzed, confirming the validity of our countermeasure implementation.

4.2 Cost of Protection

It is expected that protection against side-channel analysis will come at a price in area, performance, and power consumption. Quantifying this cost is critical for a fair evaluation of candidates since the cost of protection varies among candidates. To assess the cost of protection, we benchmarked various variants of our TinyJAMBU implementation in FPGA. The benchmarking results are listed in Table 1. Multiple variants up to the third order of protection, with a combination of N=8, 16, and 32 are shown. Each TinyJAMBU variant in Table 1 is parameterized with two parameters; d and N that correspond to the order of protection and the number of feedback bits calculated in parallel, respectively. For example, TinyJAMBU-d2-N32 is a second-order implementation that updates 32 bits in parallel. Table 1 abbreviations are Max Freq. (Maximum Frequency), TP (Throughput), TP Ratio (Throughput Ratio), LUT (Lookup Table) and Rand. Bits (Random Bits). The throughput is calculated for encryption of long messages in Mbps. We benchmarked the variants in Xilinx Artix-7 FPGA using Vivado 2020.1 for synthesis and implementation. Minerva [7] was used to search for optimized tool settings for the maximum frequency.

We further compare with results from [5], where side-channel resistant variants of NIST LWC candidates COMET-CHAM and SCHWAEMM were reported. Comparison with these results can be done directly since the implementations conform to the LWC Hardware API, and results were reported for Xilinx Artix-7 FPGA. Further, the PRNG footprint is not calculated as a part of the area. We follow the same approach since there are various trade-offs in the implementation of PRNGs, which are beyond the scope of this work. Instead of reporting the PRNG area, we provide the number of bits required to calculate the N feedback bits. Each variant in Table 1 is also compared against its unprotected baseline variant by providing the throughput and area ratios. It can be shown from Table1 that TinyJAMBU is flexible and can be efficiently protected against side-channel analysis. Also, due to its small footprint and the small size of non-linearity, one can implement high-order variants that can fit in small FPGAs. For example, the third-order TinyJAMBU-d3-N32 can easily fit in small Xilinx FPGAs such as Artix7-xc7a12tcsg325, which has 8000 LUTs. Below, we discuss our observations;

- Since we process shares in parallel, the throughput of our TinyJAMBU implementations is not significantly affected by increasing the order of protection. As shown in table 1, cycles per block remain the same. A minor reduction in maximum frequency as the order of protection increases is responsible for the throughput reduction.
- Due to the synchronization registers in the protected NAND gates, we update the state after two clock cycles. This causes the permutation to take twice the number of clock cycles compared to the unprotected variant. Pipelining may be used to reduce the number clock cycles at the expense of increasing the number of registers.

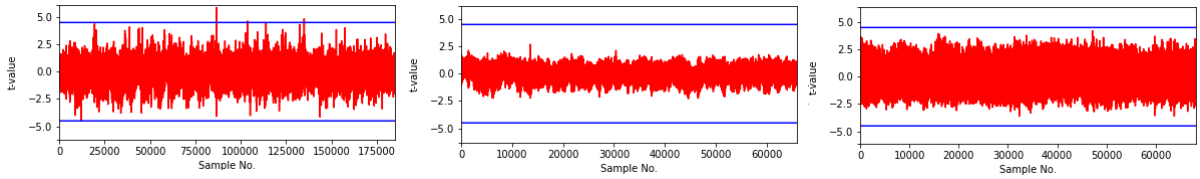

Figure 6: TVLA Results. From left to right: 1st-order TVLA on unprotected TinyJAMBU (10,000 traces) , 1st-order TVLA on 1st-order protected TinyJAMBU (1 million traces) and 2nd-order TVLA on 2nd-order TinyJAMBU (1 million traces)

Table 1: Benchmarking Results in Xilinx Artix-7 FPGA. The protection order is denoted by *d* and the number of feedback bits calculated in parallel is denoted by *N*.

Implementation	Protection Order	cycles per block	Freq. MHz	TP Mbps	TP Ratio	LUTs	LUT Ratio	Rand. Bits	Ref.
TinyJAMBU_d0-N32	unprotected	34	266	250.4	1.00	591	1.00	N/A	[13]
COMET-CHAM-d0	unprotected	-	201	282.7	4.99	2214	0.92	N/A	[5]
COMET-CHAM-KSA-d0	unprotected	-	255	56.7	1.00	2399	1.00	N/A	[5]
SCHWAEMM-d0	unprotected	-	169	920.5	3.56	2321	0.82	N/A	[5]
SCHWAEMM-KSA-d0	unprotected	-	189	258.7	1.00	2824	1.00	N/A	[5]
TinyJAMBU-d1-N32	1	66	247	119.8	0.48	1267	2.14	96	T.W.
TinyJAMBU-d1-N16	1	130	248	61.0	0.24	1086	1.84	48	T.W.
TinyJAMBU-d1-N8	1	258	249	30.9	0.12	969	1.64	24	T.W.
COMET-CHAM-KSA-d1	1	-	205	44.2	0.78	8760	3.65	-	[5]
SCHWAEMM-KSA-d1	1	-	207	231.4	0.89	12531	4.44	-	[5]
TinyJAMBU-d2-N32	2	66	237	114.9	0.46	2044	3.46	192	T.W.
TinyJAMBU-d2-N16	2	130	239	58.8	0.23	1586	2.68	96	T.W.
TinyJAMBU-d2-N8	2	258	243	30.1	0.12	1362	2.30	48	T.W.
TinyJAMBU-d3-N32	3	66	224	108.6	0.43	2842	4.81	320	T.W.
TinyJAMBU-d3-N16	3	130	229	56.4	0.23	2249	3.81	160	T.W.
TinyJAMBU-d3-N8	3	258	233	28.9	0.12	1817	3.07	80	T.W.

Table 2: COMA Resource Utilization on Artix-7 FPGA (xc7a100tcsg324-1). Table adapted from [2] except TinyJAMBU resource utilization.

Name	AES-GCM+AES-CTR			ACORN+Trivium			TinyJAMBU+Trivium		
	Slice	LUT	FF	Slice	LUT	FF	Slice	LUT	FF
AEAD_EXT	1336	3804	4432	333	1067	591	564	1765	1156
RNG	738	2352	628	241	683	460	241	683	460
Others	1024	2535	2155	1024	2535	2155	1024	2535	2155
Total w/o PUF	3098	8691	7215	1598	4285	3206	1829	4983	3771

- Due to the small size of a non-linear component, the cost of protection is low. For example, the area of the first-order variant TinyJAMBU-d1-N32 is only 2.14× the unprotected baseline variant. It is noteworthy that the control logic and constant additions in the datapath are not duplicated in the protected design.
- Our first-order TinyJAMBU-d1-N32 is only 15% the area of the first-order COMET-CHAM reported in [5], however, it can provide 2.7× the throughput.
- Although first-order SCHWAEMM-KSA is 2× faster than our first-order TinyJAMBU-d1-N32, it has roughly 10× the area. Its 12,531 LUTs size may not be suitable for small FPGAs used in lightweight applications.

4.3 Power and Energy Estimation

Power consumption and energy per bit are critical considerations for lightweight applications, especially for battery-powered devices. We provide average dynamic power and energy per bit (E/bit) estimations for Xilinx Artix7-xc7a100tftg256-3 FPGA in Table 3. We utilized Xeda [14], which provides an abstraction layer to run flows on different Electronic Design Automation (EDA) tools. In our case, Xilinx Vivado 2020.1 was run by Xeda to perform power estimation. For result accuracy, we performed vector-based estimation, and the Switching Activity Interchange format (SAIF) files were generated by encrypting 1536 bytes of plaintext using post-route timing simulation. Average dynamic power has been estimated at 10, 50, and 100 MHz. As expected, the average dynamic power and E/bit increase as the order of protection increases. For the same variant,

Table 3: Power and Energy per bit (E/bit) Estimations for Artix-7 FPGA

Variant	Freq. MHz	Avg. Dyn. Power mW	E/bit nJ/bit
TinyJAMBU-d1-N32	10MHz	5	1.0
	50MHz	24	1.0
	100MHz	47	1.0
TinyJAMBU-d2-N32	10MHz	8	1.7
	50MHz	37	1.5
	100MHz	75	1.6
TinyJAMBU-d3-N32	10MHz	12	2.5
	50MHz	59	2.5
	100MHz	119	2.5

E/bit remains approximately the same as frequency increases since the increase in power consumption is negated by a reduction in time needed to perform the operation.

4.4 Use in COMA

In Table 2, we compare the resource utilization of three different instantiations of COMA based on three authenticated ciphers AES-GCM, ACORN, and TinyJAMBU. Out of these ciphers: AES-GCM is an existing standard; ACORN has been a second choice for use case 1, Lightweight Applications, in the CAESAR cryptographic contest conducted in the period 2014-2019; and TinyJAMBU is a finalist of the NIST LWC standardization process, currently in progress.

AEAD_EXT represents an SCA-protected implementation of an authenticated cipher, RNG is a random number generator, and the row Others covers the remaining components of COMA, described in detail in [2]. The notation <Cipher 1>+<Cipher 2> means that <Cipher 1> is used for authenticated encryption and <Cipher 2> for the construction of a pseudorandom part of RNG (necessary to reach the required speed). The version of TinyJAMBU used to generate results for this table is TinyJAMBU-d1-N32, with the first-order DPA countermeasures, processing 32 bits in parallel. The obtained results clearly demonstrate that the solution based on TinyJAMBU+Trivium outperforms the solution based on existing standards, AES-GCM+AES-CTR, in terms of resource utilization. In particular, for AEAD_EXT, the usage of LUTs is reduced by a factor of 2.2 and the usage of flip-flops by a factor of 3.8. When all other components, except a PUF, are taken into account, the corresponding ratios are 1.7 and 1.9. The solution based on ACORN+Trivium has slightly smaller resource utilization, but ACORN is not any longer considered for standardization due to the security concerns and long initialization time. As a result, TinyJAMBU emerges as a strong contender for protecting obfuscation keys during secure activation of integrated circuits.

5 CONCLUSIONS AND FUTURE WORK

In this work, we present a design space exploration study for side-channel resistant hardware implementations of TinyJAMBU. To achieve this goal, we implemented a flexible design that can be synthesized for an arbitrary order of protection. The implementations can trade performance for the area by configuring the level of parallelism. This design with two degrees of freedom is flexible and can be easily adapted for applications with different security, cost, and performance trade-offs. The security of our implementations has been verified up to the second order using the TVLA methodology. The benchmarking results on Xilinx Artix7 FPGA show that one can instantiate high-order implementations of TinyJAMBU in small FPGAs. Concretely, our third-order TinyJAMBU-d3-N32 variant is only 2842 LUTs and can easily fit in small FPGAs. Our conclusion is that TinyJAMBU is suitable for producing lightweight side-channel-resistant implementations of authenticated encryption. In particular, it can be used for high-speed and low-overhead protection of obfuscation keys in hardware security applications. The evaluation of other protection schemes and quantifying the associated cost and performance will be interesting for future work.

REFERENCES

[1] Abubakr Abdulgadir, William Diehl, and Jens-Peter Kaps. 2019. An Open-Source Platform for Evaluation of Hardware Implementations of Lightweight Authenticated Ciphers. In *2019 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*. IEEE, Cancun, Mexico, 1–5. https://doi.org/10.1109/ReConFig48160.2019.8994788

[2] Kimia Zamiri Azar, Farnoud Farahmand, Hadi Mardani Kamali, Shervin Roshanisefat, Houman Homayoun, William Diehl, Kris Gaj, and Avesta Sasan. 2019. COMA: Communication and Obfuscation Management Architecture. In *22nd International Symposium on Research in Attacks, Intrusions and Defenses*. USENIX Association, Beijing, China, 181–195.

[3] CERG - GMU. [n.d.]. GMU Cryptographic Engineering Group GitHub. https://github.com/GMUCERG/.

[4] Suresh Chari, Charanjit S Jutla, Josyula R Rao, and Pankaj Rohatgi. 1999. Towards Sound Approaches to Counteract Power-Analysis Attacks. In *CRYPTO*. 15. https://doi.org/10/fr3jwj

[5] Flora Coleman, Behnaz Rezvani, Sachin Sachin, and William Diehl. 2020. Side Channel Resistance at a Cost: A Comparison of ARX-Based Authenticated Encryption. In *2020 30th International Conference on Field-Programmable Logic and Applications (FPL)*. IEEE, Gothenburg, Sweden.

[6] Christophe De Canniere and Bart Preneel. 2005. *TRIVIUM Specifications*. Technical Report.

[7] Farnoud Farahmand, Ahmed Ferozpuri, William Diehl, and Kris Gaj. 2017. Minerva: Automated Hardware Optimization Tool. In *2017 International Conference on ReConFigurable Computing and FPGAs, ReConFig 2017*. IEEE, Cancun, 1–8.

[8] Gilbert Goodwill, Benjamin Jun, Josh Jaffe, and Pankaj Rohatgi. 2011. A Testing Methodology for Sidechannel Resistance Validation. In *NIST Non-Invasive Attack Testing Workshop*.

[9] Hannes Groß. 2018. *Domain-Oriented Masking - Generically Masked Hardware Implementations*. PhD Thesis. Graz University of Technology, Austria.

[10] Hannes Gross, Stefan Mangard, and Thomas Korak. 2016. *Domain-Oriented Masking: Compact Masked Hardware Implementations with Arbitrary Protection Order*. Cryptology ePrint Archive 2016/486.

[11] Paul Kocher, Joshua Jaffe, and Benjamin Jun. 1999. Diferential Power Analysis. In *CRYPTO '99 - 19th International Conference on Cryptology*. Santa Barbara, CA.

[12] Stefan Mangard, Thomas Popp, Berndt M. Gammel, Stefan Mangard, Thomas Popp, and Berndt M. Gammel. 2005. Side-Channel Leakage of Masked CMOS Gates. In *Topics in Cryptology – CT-RSA 2005 (LNCS, Vol. 3376)*. Springer, Berlin, Heidelberg, 351–365. https://doi.org/10.1007/978-3-540-30574-3_24

[13] Kamyar Mohajerani, Richard Haeussler, Rishub Nagpal, Farnoud Farahmand, Abubakr Abdulgadir, Jens-Peter Kaps, and Kris Gaj. 2020. *FPGA Benchmarking of Round 2 Candidates in the NIST Lightweight Cryptography Standardization Process: Methodology, Metrics, Tools, and Results*. Cryptology ePrint Archive 2020/1207.

[14] Kamyar Mohajerani and Rishub Nagpal. 2020. Xeda. https://github.com/kammoh/xeda.

[15] Svetla Nikova, Christian Rechberger, and Vincent Rijmen. 2006. Threshold Implementations Against Side-Channel Attacks and Glitches. In *Information and Communications Security, ICICS 2006 (LNCS, Vol. 4307)*. Springer Berlin Heidelberg, 529–545. https://doi.org/10.1007/11935308_38

[16] Tobias Schneider, Amir Moradi, Tobias Schneider, and Amir Moradi. 2015. Leakage Assessment Methodology: A Clear Roadmap for Side-Channel Evaluations. In *Cryptographic Hardware and Embedded Systems – CHES 2015*. Springer Berlin Heidelberg, Berlin, Heidelberg, 495–513.

[17] Hongjun Wu and Tao Huang. 2019. TinyJAMBU: A Family of Lightweight Authenticated Encryption Algorithms. *Submission to the NIST Lightweight Cryptography Standardization Process* (March 2019).