ExTru: A Lightweight, Fast, and Secure *Ex*pirable *Tru*st for the Internet of Things

Hadi M Kamali*, Kimia Z Azar*, Shervin Roshanisefat*, Ashkan Vakil*, Houman Homayoun†, Avesta Sasan*
*Department of ECE, *George Mason University*, Fairfax, VA, USA
{hmardani, kzamiria, sroshani, avakil, asasan}@gmu.edu
†Department of ECE, *University of California, Davis*, Davis, CA, USA
hhomayoun@ucdavis.edu

2020 IEEE 14th Dallas Circuits and Systems Conference (DCAS) | 978-1-7281-8510-1/20/$31.00 ©2020 IEEE | DOI: 10.1109/DCAS51144.2020.9330632

***Abstract*—The resource-constrained nature of the Internet of Things (IoT) edges, poses a challenge in designing a secure and high-performance communication for this family of devices. Although side-channel resistant ciphers (either block or stream) could guarantee the security of the communication, the *energy*-intensive nature of these ciphers makes them undesirable for lightweight IoT solutions. In this paper, we introduce *ExTru*, an encrypted communication protocol based on stream ciphers that adds a configurable switching & toggling network (*CSTN*) to not only boost the performance of the communication in these devices, it also consumes far less energy than the conventional side-channel resistant ciphers. Although the overall structure of the proposed scheme is leaky against physical attacks, we introduce a dynamic encryption mechanism that removes this vulnerability. We demonstrate how each communicated message in the proposed scheme reduces the level of trust. Accordingly, since a specific number of messages, N, could break the communication and extract the key, by using the dynamic encryption mechanism, *ExTru* can re-initiate the level of trust periodically after T messages where $T < N$, to protect the communication against side-channel and scan-based attacks (e.g. SAT attack). Furthermore, we demonstrate that by properly configuring the value of T, *ExTru* not only increases the strength of security from per "*device*" to per "*message*", it also significantly improves energy saving as well as throughput vs. an architecture that only uses a conventional side-channel resistant block/stream cipher.**

***Index Terms*—Internet-of-Thing, Secure Communication, Block Cipher, Stream Cipher, Physical Attack**

I. Introduction

The Internet of Things (IoT), which has been foreseen to become the most successful business for the next decade by *International Technology Roadmap for Semiconductors* (ITRS), is an inevitable landmark of smart life providing novel applications and services, ranging from business automation to personal day-to-day life [1]. The IoT infrastructure is the seamless connection of billions of heterogeneous devices (*"things"*) within a large integrated network (the *"Internet"*). The heterogeneity of IoT constitutes from a wide variety of devices, such as smartwatches, mobile phones, etc, which results in a drastic increase in the number of IoT devices.

Although IoT devices provide a more efficient, automated, and smart life, from security/privacy perspective, many threats and vulnerabilities have been raised in IoT devices. Many investigations on cyber-based threats demonstrate that there are 176 new cyber-threats every minute, and over 2.5 million within only four months [2]. Several incidents have highlighted the massive influence of counterfeit/cloned/tampered devices into the supply chain [3]. As an instance, influencing and controlling every connected device within a ZigBee network, which is one of the most prevalent wireless communications in IoT, has been illustrated in [4], [5]. Another recent evaluation by HP demonstrates that 70% of the devices in IoT are vulnerable to different types of threats, including physical attacks.

Numerous solutions, including communication standards optimization, more secure configuration, etc, have been introduced to protect IoT devices and their communications against physical threats, which help to prevent the wide variety of conventional attacks [7]. For instance, the utilization of symmetric-based secret-key ciphers or keyed hash-based authentication code (HMAC) is prevalent in IoT devices to provide integrity and authentication while securely protect the inter-communication of IoT edge devices. Considering that the power consumption (particularly energy consumption) constraints in resource-constrained edge devices are very strict, the energy overhead of security solutions against hardware threats must be minimized. For instance, tight restrictions in edge devices enforce the designer to employ lightweight ciphers, such as stream ciphers or lightweight block ciphers [8], [9]. However, the energy consumption of this breed of encryption architectures is still high for a high portion of IoT edge devices. Also, the performance of these ciphers considerably lower than regular block ciphers. This creates an inevitable security/cost trade-off in lightweight IoT devices, which results in sacrificing one of them.

In this paper, we introduce a new lightweight, fast, and provably secure *Exp*irable *Tr*ust (*ExTru*) mechanism relied on a configurable switching and toggling network (CSTN) as well as the winner of the Competition for Authenticated Encryption: Security, Applicability, and Robustness (CAESAR) [10], called ACORN [11]. *ExTru* provably protects the inter-communication of IoT edge devices while it even obtains higher performance and mitigates the energy consumption compared to the case in which the regular block/stream ciphers have been used. Moreover, we show how *ExTru* engages dynamicity in the circuit to provide guaranteed protection against different types of physical and scan-based attacks, such as side-channel, Boolean satisfiability (SAT) attack, and algebraic attack. We demonstrate that by using this dynamic encryption scheme, the strength of security could be elevated from *per device* to *per message*.

978-1-7281-8510-1/20/$31.00 ©2020 IEEE

II. Related Work

Due to the resource-constrained nature of IoT devices, a big challenge in guaranteeing the security of this group of devices is that the implementation of the security measures must be sufficiently lightweight, which prevents the designers to directly use conventional block ciphers [12], [13], such as AES-GCM [13]. Many studies have been taken by the research community to not only address security issues in IoT networks but also to increase the efficiency by lowering the power (particularly energy) consumption and increasing the throughput. For instance, the fact that the elliptic curves cryptography (ECC) achieves guaranteed security with reduced resource requirements has attracted the research community. The work in [14] has constructed an optimized ECC for secure communication in heterogeneous IoT devices based on Schnorr signature. Also, a simple key negotiation protocol has been introduced in this work that is based on the Schnorr scheme to demonstrate the usability of the presented ECC optimizations.

Based on the desirable features of a physically unclonable function (PUF), such as lightweightedness, unpredictability, unclonability, and uniqueness, many researchers have been motivated to concentrate on the usage of this module to build a secure communication for IoT devices. Among several studies on PUF-based secure communication for IoT devices, the work in [15] has introduced an authentication, key sharing, and secure communication architecture, in which each IoT device has an integrated PUF. In this work, the identity of each device is created by the challenge-response pair signature of its PUF instance, and by engaging the identity-based encryption scheme proposed in Boneh and Franklin, the security of this approach is proven against attacks like chosen-plaintext/ciphertext attack.

Numerous software/hardware implementation of lightweight ciphers suited for IoT devices have been proposed in recent few years, including RECTANGLE, PICO, Extended-LILIPUT, SIT, SKINNY, MANTIS, to name but a few [16]. Some of these ciphers could provide the best performance on software implementation, however, a portion of them have better performance in hardware implementation. For instance, the work in [17] introduces a lightweight 64-bit symmetric block cipher, called SIT, whose implementation is a mixture of Feistel and a uniform substitution-permutation network. The proposed approach uses some logical operations along with some swapping and substitution. Most of the encryption algorithms designed for IoT reduced the number of rounds to make a cost-security trade-off. For instance, SIT uses five rounds of encryption with 5 different keys to improve energy efficiency.

The lightweightedness of the stream ciphers, on the other hand, has received fascinated attention from many researchers' in recent years [18]. Since IoT being an emerging field requires lightweight cipher designs with robustness, less complexity, and lower energy consumption, stream ciphers are very suited for particularly edge devices. Many studies evaluate the possibility of engaging stream ciphers in IoT devices, such as WG-8, Trivium, Quavium, and ACORN [18].

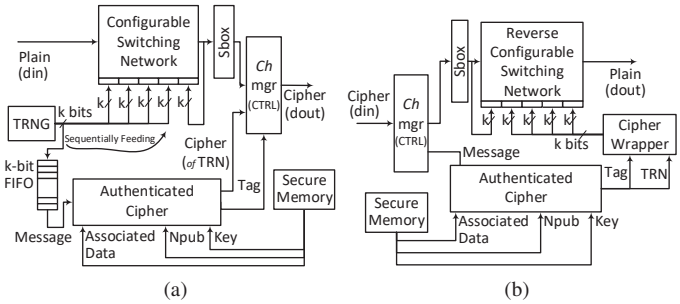


Fig. 1. *ExTru* Overall Infrastructure (a) Transmitter, (b) Receiver

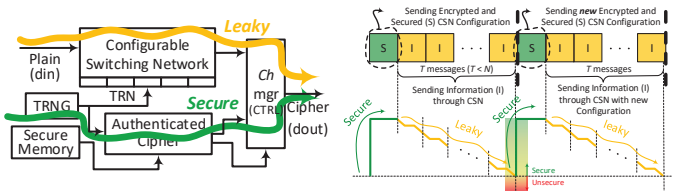
III. *ExTru* Infrastructure

ExTru consists of four main sub-modules: (1) ACORN as a stream cipher that would be used periodically (The frequency will be discussed further), (2) a configurable switching and toggling network (CSTN) that dynamically permutes/toggles the data based on the configuration generated by TRNG, (3) a random number generator (RNG) that is responsible for generating random data for Threshold Implementation of ACORN as well as for generating the CSTN configuration, and (4) a substitution box placed after CSTN to eliminate the linearity/predictability of the ciphertext. The overall architecture of *ExTru* has been demonstrated in Fig. 1 for both transmitter side and receiver side.

On the transmitter side, the CSTN is used to permute/toggle the plaintext using the configuration (TRN) generated by the random number generator (RNG). The RNG will periodically change the configuration (TRN) to add dynamicity into the permutation/toggle network (CSTN). Parts of the configuration is fed by the permuted/toggled data (the output of the CSTN) to make the operation stateful (data-dependent). The CSTN is followed only by a substitution-box to eliminate the linearity/predictability of the output. The TRN that is used to configure the CSTN has been also encrypted using the authenticated cipher to be transmitted to the receiver. The key used for authenticated cipher could be pre-stored in the secure memory or produced by a PUF. The output of the transmitter (ciphertext) would be selected from the output of the s-box (permuted/toggled + substituted plaintext) or authenticated cipher output (encrypted TRN).

On the receiver side, on the other hand, the reverse CSTN (RCSTN) must be used to recover the permuted/toggled + substituted plaintext. We will show that similar to ACORN that engages only one hardware module for both encryption/decryption, the CSTN hardware is the same for both receiving/sending operations (same hardware for both CSTN and RCSTN). Hence, no duplicated hardware (one for CSTN and one for RCSTN) is required to be added on each side. When TRN is received from the transmitter it must be decrypted using the authenticated cipher to be used as the configuration of the RCSTN. If the received data is not TRN, it first must pass the s-box to accomplish re-substitution, then it must pass the RCSTN to recover the plaintext.

Fig. 2 depicts the overall structure of dynamic encryption provided by *ExTru*, which has no sign of leaky communication. As shown in Fig. 2(b), for each specific number of



(a) (b)

Fig. 2. (a) The overall structure of using Dynamically Encrypted communication (b) Re-intensifying the Security Strength by Dynamically Encrypted communication

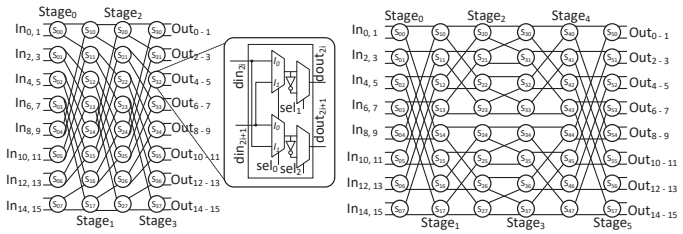
transmission (T), which must be less than N, a new CSTN configuration will be sent via side-channel resistant cipher. As it is shown, a secure message (S), which contains TRN, will be sent periodically after every T messages (I) that are handled by CSTN/RCSTN. Based on different forms of attacks, such as side-channel, scan-based SAT, and algebraic attack, messages (I) are leaky. So, periodically changing TRN (S) and sending through side-channel resistant ciphers re-intensify the security of the communication.

Based on the size of the CSTN/RCSTN (number of I/O), we will show that the maximum feasible update frequency (N) would be changed. Consequently, the CSTN configuration (TRN), which is fed by RNG, must be changed dynamically after every T iterations, where $T < N$. Also, the size of CSTN/RCSTN determines the number of configuration bits (size of each S) must be generated by the RNG. In the following sub-sections we discuss the details of *ExTru* implementation.

A. Configurable Switching & Toggling Network (CSTN)

The CSTN is a logarithmic routing (permutation) network that could permute the order of the signals at its input pins to its output pins while possibly toggling their logic levels based on its configuration (TRN). Fig. 3(a) captures a simple implementation of an 8×8 CSTN based on *OMEGA* network [19]. The network is constructed using permutation elements, denoted as Re-Routing Blocks (RRB). Each RRB is able to possibly toggle and permute each of the input signals to each of its outputs. The number of RRBs needed to implement this simple CSTN for N inputs (N is a power of 2) is simply $N/2 \times logN$.

Each CSTN should be paired with an RCSTN. RCSTN must be able to reverse all operations accomplished by CSTN to re-generate the plaintext. Due to the structure of CSTN, RCSTN can be implemented by *vertically flipping* the CSTN without any change in configuration [21]. However, to avoid duplicating the hardware (to put one dedicated hardware for CSTN and one dedicated hardware for RCSTN), by flipping the configuration bits (row-pivot reversed TRN), the CSTN would operate as its corresponded reverse CSTN. Hence, only one hardware is required to operate as both CSTN and RCSTN. In *ExTru*, we engage a near non-blocking CSTN which are more resilient against state-of-the-art attacks [22]–[25].



(a) Blocking *OMEGA* (b) Near Non-blocking $LOG_{8,1,1}$

Fig. 3. Logarithmic Network (a) Blocking, (b) Near Non-blocking.

B. Authenticated Encryption with Associated Data

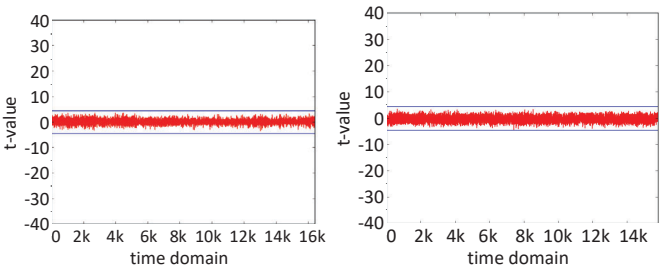
The Authenticated Encryption with Associated Data (AEAD) is used in *ExTru* for the transmission of the CSTN-RCSTN configuration (TRN). Authenticated ciphers incorporate the functionality of confidentiality, integrity, and authentication. The input of an authenticated cipher includes *plaintext* (message), *associated data* (AD), *public message number* (NPUB), and *secret key*. Then, the *ciphertext* is generated as a function of these inputs. A *tag*, which depends on all inputs, is generated after message encryption to assure the integrity and authenticity of the transaction. This tag is then verified after the decryption process. The choice of AEAD could significantly affect the area overhead of the solution, the speed of encrypted communication, and the extra energy/power consumption. To show the performance, power/energy, and area trade-offs, we employ two AEAD solutions: a NIST compliant solution (AES-GCM) [13], and a promising lightweight solution (ACORN) [11]. Also, in both versions, threshold implementation (TI) has been engaged to resist against side-channel attacks [26].

C. Random Number Generator (RNG)

A RNG unit is required on both sides to generate random bits for side-channel protection of AEAD units, a random public message number (NPUB) for AEAD, and TRNs for CSTN-RCSTN. We adopted the ERO TRNG core described in [27], which is capable of generating only 1-bit of random data per over 20,000 clock cycles. In our TI implementations, AES-GCM needs 40 and ACORN 15 bits of random data per cycle. So, we employed a hybrid RNG unit combining the ERO TRNG with a Pseudo Random Number Generator (PRNG). TRNG output is used as a 128-bit seed to PRNG. The PRNG generates random numbers needed by other components. The reseeding is performed only once per activation. PRNG is implemented in two different versions: (1) AES-CTR PRNG, which is based on AES, and (2) Trivium based PRNG, which is based on the Trivium stream cipher.

D. Substitution Box (S-Box)

To eliminate the linearity/predictability in *ExTru*, a non-feistel trial strategy has been used that is based on Khazad block cipher [28]. The wide trial strategy is composed of several linear and non-linear transformations that ensures the dependency of output bits on input bits in a complex manner [29].



(a) AES-GCM protected (b) ACORN protected

Fig. 4. The t-test results for the protected implementations AEADs.

IV. Security Analysis of *ExTru*

Assuming that the attacker can monitor the side-channel information of the chips during normal operations (based on power/current traces), and there exist no secure scan chain structure, e.g. [30], [31], to block the access to the scan chain, in this section we evaluate the resiliency of *ExTru* against different physical attacks, such as side-channel, the scan-based SAT, and algebraic attack. An Attack objective may be (1) extracting the secret key, or (2) extracting CSTN configuration (TRNs), or (3) eavesdropping on messages exchanged between the devices.

A. Side-Channel Attack (SCA)

The objective of SCA on *ExTru* is to extract either the secret key used by AEAD (ACORN) or the TRN used by CSTN. It is worth mentioning that assuming that the secret key or TRN is extracted, the functionality of the s-box would be revealed using specific messages. Fig. 4 captures our assessment of the side-channel resistance of AEAD using a t-test for the protected implementations of AES-GCM and ACORN [32]. As illustrated, both implementations pass the t-test, indicating the guaranteed resistance against SCA. Note that this guaranteed resistance against SCA shows the robustness of communication channel during TRN transmission.

B. TRN Extraction using the SAT attack

Since the attacker might have access to the scan chain to apply any form of scan-based attack, it might be possible to recover and extract the TRN by applying specific inputs to the CSTN and observing the output. This could be done by using the SAT attack that is a very applicable and known attack on logic locking schemes [33], [34]. In this scheme, assuming that the TRN is the unknown parameters (such as key in logic locking), based on Table I, it is evident that using near non-blocking CSTN considerably enhances the resiliency of this approach against the SAT attack. As shown in Table I, for a near non-blocking CSTN with a size of 64 ($LOG_{64,4,1}$), the SAT is not able to find the TRN after 2×10^6 seconds. Even after 2×10^6 seconds execution of SAT, it cannot find more than 5 DIPs. However, based on the SAT iterations for $LOG_{32,3,1}$, we expect that for a close to non-blocking CSTN with size 64, more than 32 DIPs are required to extract CSTN configuration.

TABLE I
SAT Execution Time on a Close to Non-blocking CSTN, $LOG_{n,log_2(n)-2,1}$, for Different Sizes.

CSTN Size (n)	4	8	16	32	64
SAT Iterations	14	18	25	32	TO
SAT Execution Time $_{(Seconds)}$	0.01	0.015	2.35	79.18	TO

TO: Timeout = 2×10^6 *seconds*

TABLE II
Main Features of the two ExTru Modes.

Feature	Block	Stream
AEAD	AES-GCM	ACORN
PRNG	AES-CTR	Trivium
BUS Width	8	8
Pins used for Communication	8	8
CSTN-RCSTN Size	64	64
Trusted Memory	4 Kbits	4 Kbits
C_{fix}: initialization overhead (cycles)	10,492	20,452
C_{byte}: cycles needed for encrypting 1 byte	72	17
$PRNG_{perf}$: Throughput of generating TRN	128b/10cycles	64bit/cycle

C. Algebraic Attacks

CSTN can be expressed as an affine function of the data input x, of the form $y = A \cdot x + b$, where A is an $n \times n$ matrix and b is an $n \times 1$ vector, with all elements dependent on the input TRN. Although recovering A and b is not equivalent to finding the TRN, it may enable the successful decryption of all blocks encrypted using a given TRN. We protect against this threat in numerous ways: (1) The number of blocks encrypted using a given TRN is set to the value smaller than n, which prevents generating and solving a system of linear equations with A and b treated as unknowns, (2) a part of the configuration is data-dependent and is fed from the output of the CSTN (stateful), so the values of A and b are not the same in any two encryptions, without the need of feeding CSTN with two completely different TRN values, (3) the substitution box added after the CSTN will eliminate all linearity/predictability of the CSTN using the algebraic attack.

V. Experimental Setup and Analysis

For evaluation, all designs have been implemented using Verilog HDL, and have been synthesized for both FPGA and ASIC targets. For ASIC verification, we used Synopsys generic 32nm process. For FPGA verification, we targeted a small FPGA board, Digilent Nexys-4 DDR with Xilinx Artix 7 (XC7A100T-1CSG324). In addition, for SAT evaluation, we employed the Lingling-based SAT attack [35] on a Dell PowerEdge R620 equipped with Intel Xeon E5-2670 2.6 GHz and 64GB of RAM. Also, as noted, a run-time limit of 2×10^6 seconds was set for the SAT solver. For ciphers, we used two side-channel resistant ciphers (AES-GCM128 as a block authenticated cipher, and ACORN as a lightweight stream cipher). We have two modes in *ExTru*: (1) *ExTru* with AES-GCM, compared with its corresponding cipher (AES-GCM), (2) *ExTru* with ACORN, compared with its corresponding cipher (ACORN). All configurations are listed in Table II.

Table III demonstrates the overhead of $LOG_{64,4,1}$ compared to both ciphers using Synopsys generic 32nm library, after

TABLE III
POWER, AREA, AND DELAY OF $LOG_{64,4,1}$ COMPARED TO PROTECTED AES-GCM AND ACORN

Design	Power (uW)	Area (nm^2)	Delay (ns)
$LOG_{64,4,1}$	1625.5	9965.9	1.74
AES-GCM	3587.1	102487.5	2.48
ACORN	880.9	21843.4	2.3

TABLE IV
SAT EXECUTION TIME ON CLOSE TO NON-BLOCKING CSTN, $LOG_{n,log_2(n)-2,1}$, FOR DIFFERENT SIZES.

CSTN	Area (nm^2)	Power (uW)	Delay (ns)	SAT-Resilient
omega32	1013.1	44.8	1.12	✗
log(32, 3, 1)	3067.5	213.5	1.33	✗
omega64	2285.5	107.1	1.22	✗
log(64, 4, 1)	**7438.8**	**845.1**	**1.73**	✓
omega128	5081.5	250.3	1.25	✗
omega256	11364.9	579.1	1.35	✗
omega512	**25458.3**	**2308**	**1.42**	✓

post-layout (route) verification (PLS). As it can be seen, PLS reports show that the power consumption of $LOG_{64,4,1}$ is higher than ACORN. However, based on the area utilization, $LOG_{64,4,1}$ is considerably smaller than ACORN and AES-GCM. The main reason is that the switching activity of CSTN is high due to numerous permutation/toggling + substitution which leads to have higher power consumption than ACORN. Additionally, the delay of critical paths in both ciphers is higher than that of CSTN. Based on Fig. 1, it is obvious that critical path in *ExTru* is same as that of its corresponding cipher. Consequently, we expect that the delay of critical path in *ExTru* is approximately equal with that of ciphers.

Also, Table IV depicts area, power, and the delay of CSTNs in both blocking and near non-blocking mode with different sizes in the Synopsys generic 32nm process. As shown, it is evident that using a close to non-blocking CSTN with size 64, $LOG_{64,4,1}$, provides the most efficient CSTN structure, which is resilient against SAT attack. It should be noted that due to having extra stages in close to non-blocking CSTNs, the delay of these networks is slightly higher than the blocking CSTNs with the same n, which is negligible.

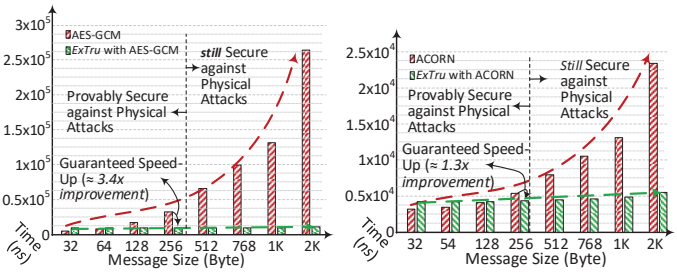
Table V depicts resource utilization of *ExTru* in each mode of using AES-GCM or ACORN. As we expected, the critical paths of *ExTru* in each mode is same as that of corresponding cipher. In addition, since *ExTru* consists of both CSTN and cipher, it is evident that area and power of *ExTru* in each mode is approximately equal to summation of total area and total power of both sub-modules, i.e. CSTN and the cipher.

A. Energy/Performance Improvement in ExTru

Although combining CSTN and cipher into *ExTru* imposes area and power overhead by almost 24.5% compared to the corresponding cipher, CSTN can generate {permuted/toggled + substituted} data in only one cycle which provides significant speed-up compared to especially side-channel resistant ciphers that require randomness or complex initialization. Fig. 5 demonstrates the time of preparing data (encryption *or* permutation/toggling + substitution) for different message sizes. Increasing the size of the message, which increases the

TABLE V
ExTru RESOURCE UTILIZATION WITH DIFFERENT CIPHERS [32]

Design	Power (uW)	Area (nm^2)	Delay (ns)
ExTru with AES-GCM	4448.9	122457.4	2.48
ExTru with ACORN	1694.6	33344.7	2.3



(a) (b)

Fig. 5. The Time of Preparation Data (Encryption *or* Permutation/Toggling + Substitution) for Different Message Sizes (a) AES-GCM vs. *ExTru* with AES-GCM (b) ACORN vs. *ExTru* with ACORN.

proportion of I to S, significantly (superlinearly) increases the gap between the execution time of *ExTru* compared to its corresponding cipher. As shown, since CSTN prepares each I in one cycle, increasing the size of the message imposes no degradation on *ExTru* performance. The main part of the execution time of *ExTru* is dedicated to encrypting and sending S. On the other hand, all data must be encrypted before sending it while only a cipher is used. So, it increases the execution time of ciphers linearly due to encryption time. Note that based on our SAT-based evaluation, the guaranteed number of I messages is 32 (Table I). Since we use $LOG_{64,4,1}$, each I is 64 bits, so 256B (64×32 = 2Kb = 256B) is the safe size of sending data through CSTN. The guaranteed speed-up is $3.4\times$ and $1.3\times$ compared to AES-GCM and ACORN.

It is evident that for small messages, *ExTru* works slower than ciphers due to time overhead of sending encrypted TRN. However, *ExTru* can accelerate the execution time up to $25\times$ while the message size is even 2KB. The speed-up gained by *ExTru* depends on the structure of the cipher. For instance, the AES-GCM needs around 300 cycles per each plain data to be first-order side-channel resistant. However, ACORN as a stream cipher needs fewer cycles per data. So, *ExTru* provides better speed-up while the cipher is not streamed/pipelined.

Table VI depicts energy consumption for different designs, with different message sizes. Since energy is a function of time and power, it is obvious that the energy consumption in *ExTru* is higher for small message sizes due to the time overhead of sending encrypted TRN. However, increasing the size of the network results in significantly less energy consumption in *ExTru* compared to corresponding ciphers. As it can be seen, *ExTru* reduces energy consumption by 94.5% and 67.8% compared to GCM and ACORN, respectively.

As mentioned previously, *ExTru* has been verified on both ASIC and FPGA. Table VII demonstrates the resource utilization of the proposed scheme compared to ciphers on Nexys-4 DDR with Xilinx Artix 7. The results in FPGA are approximately similar to that of ASIC. As expected, ACORN provides higher maximum frequency due to its lightweight structure.

TABLE VI
THE ENERGY CONSUMPTION (ENCRYPTION *or* PERMUTATION/TOGGLING + SUBSTITUTION) FOR DIFFERENT MESSAGE SIZES.

Design \ Size*	32B	64B	128B	256B	512B	768B	1KB	2KB
ACORN	17.01	18.69	22.06	28.79	42.26	55.73	69.20	123.1
ExTru with ACORN	30.66	30.80	31.09	31.67	32.82	33.98	35.13	39.75
AES-GCM	46.28	93.28	188.2	379.8	756.6	1143	1523	3055
ExTru with AES-GCM	151.1	151.4	152.1	153.3	155.8	158.4	160.9	173.1

* *Message Size*

TABLE VII
RESOURCE UTILIZATION OF *ExTru* COMPARED TO CORRESPONDING CIPHERS IN NEXYS-4 DDR WITH XILINX ARTIX 7 (XC7A100T-1CSG324).

Design	LUTs	Registers	Maximum Frequency
ACORN	1090	530	178.5 MHz
ExTru with ACORN	1609	1573	172.5 MHz
AES-GCM	3803	4418	158.3 MHz
ExTru with AES-GCM	4376	5461	152.4 MHz

However, using more resources in high-performance AES-GCM results in better throughput even with lower frequency.

VI. CONCLUSION

In this paper, we proposed *ExTru* as a dynamic encrypted high speed communication, which is able to provide a level of trust using near non-blocking configurable switching and toggling network (*CSTN*). *ExTru* uses near non-blocking CSTN as a transceiver data. Although the configuration of CSTN will be generated by TRNG, *ExTru* changes the configuration based on a time-interval which is identified by the SAT to guarantee the security of communication. Using this dynamically encrypted mechanism mitigates energy consumption by 94.5% and 67.8% compared to AES-GCM (authenticated) and ACORN (stream) while security is guaranteed. In addition, *ExTru* is able to provide up to $24.4\times$ and $4.3\times$ speed-up for 2KB messages in comparison with AES-GCM and ACORN.

ACKNOWLEDGEMENT

This research is supported by National Science Foundation (NSF, #1718434).

REFERENCES

[1] S. Li *et al.*, "The Internet of Things: A Survey," *Information Systems Frontiers*, vol. 17, no. 2, pp. 243–259, 2015.
[2] M. Frustaci *et al.*, "Evaluating Critical Security Issues of the IoT World: Present and Future Challenges," *IEEE Internet of things journal*, vol. 5, no. 4, pp. 2483–2495, 2017.
[3] M. Rostami *et al.*, "A Primer on Hardware Security: Models, Methods, and Metrics," *IEEE Proceedings*, vol. 102, no. 8, pp. 1283–1295, 2014.
[4] E. Ronen *et al.*, "IoT Goes Nuclear: Creating a ZigBee Chain Reaction," in *IEEE Symposium on Security and Privacy (SP)*, 2017, pp. 195–212.
[5] H. M. Kamali *et al.*, "MUCH-SWIFT: A High-Throughput Multi-Core HW/SW Co-design K-means Clustering Architecture," in *Great Lakes Symposium on VLSI (GLSVLSI)*, 2018, pp. 459–462.
[6] K. Z. Azar *et al.*, "NNgSAT: Neural Network guided SAT Attack on Logic Locked Complex Structures," *Int'l Conference on Computer-Aided Design (ICCAD)*, pp. 1–9, 2020.
[7] J. Yuan *et al.*, "A Reliable and Lightweight Trust Computing Mechanism for IoT Edge Devices based on Multi-Source Feedback Information Fusion," *IEEE Access*, vol. 6, pp. 23 626–23 638, 2018.
[8] D. Dinu *et al.*, "Triathlon of Lightweight Block Ciphers for the Internet of Things," *Journal of Cryptographic Engineering*, vol. 9, no. 3, pp. 283–302, 2019.
[9] R. Beaulieu *et al.*, "SIMON and SPECK: Block Ciphers for the Internet of Things," *IACR Cryptology ePrint Archive*, vol. 2015, p. 585, 2015.
[10] CAESAR, "Competition for Authenticated Encryption: Security, Applicability, and Robustness," 2013.
[11] H. Wu, "ACORN: A Lightweight Authenticated Cipher (v3)," *Candidate for the CAESAR Competition. See also https://competitions. cr. yp. to/round3/acornv3. pdf*, 2016.
[12] H. M. Kamali *et al.*, "A fault tolerant parallelism approach for implementing high-throughput pipelined advanced encryption standard," *JCSC*, vol. 25, no. 09, p. 1650113, 2016.
[13] M. J. Dworkin, *SP 800-38D. Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*. National Institute of Standards & Technology, 2007.
[14] L. Marin *et al.*, "Optimized ECC Implementation for Secure Communication between Heterogeneous IoT Devices," *Sensors*, vol. 15, no. 9, pp. 21 478–21 499, 2015.
[15] U. Chatterjee *et al.*, "A PUF-based Secure Communication Protocol for IoT," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 16, no. 3, pp. 1–25, 2017.
[16] D. Sehrawat *et al.*, "Lightweight Block Ciphers for IoT based Applications: A Review," *Int'l Journal of Applied Engineering Research*, vol. 13, no. 5, pp. 2258–2270, 2018.
[17] M. Usman *et al.*, "SIT: A Lightweight Encryption Algorithm for Secure Internet of Things," *arXiv preprint arXiv:1704.08688*, 2017.
[18] C. Manifavas *et al.*, "A Survey of Lightweight Stream Ciphers for Embedded Systems," *Security and Communication Networks*, vol. 9, no. 10, pp. 1226–1246, 2016.
[19] H. Ahmadi *et al.*, "A Survey of Modern High-Performance Switching Techniques," *IEEE Journal on Selected Areas in Communications*, vol. 7, no. 7, pp. 1091–1103, 1989.
[20] H. M. Kamali *et al.*, "LUT-lock: A novel LUT-based logic obfuscation for FPGA-bitstream and ASIC-hardware protection," in *IEEE Symposium on VLSI (ISVLSI)*, 2018, pp. 405–410.
[21] L. R. Goke *et al.*, "Banyan networks for partitioning multiprocessor systems," in *Int'l Symposium on Comp. Arch. (ISCA)*, 1973, pp. 21–28.
[22] H. M. Kamali *et al.*, "Full-lock: Hard Distributions of SAT Instances for Obfuscating Circuits using Fully Configurable Logic and Routing Blocks," in *Design Automation Conference (DAC)*, 2019, pp. 1–6.
[23] K. Z. Azar *et al.*, "COMA: Communication and Obfuscation Management Architecture," in *Int'l Symposium on Research in Attacks, Intrusions and Defenses (RAID)*, 2019, pp. 181–195.
[24] H. M. Kamali *et al.*, "SCRAMBLE: The State, Connectivity and Routing Augmentation Model for Building Logic Encryption," *IEEE Symposium on VLSI (ISVLSI)*, pp. 1–7, 2020.
[25] H. M. Kamali *et al.*, "InterLock: An Intercorrelated Logic and Routing Locking," *Int'l Conference on Computer-Aided Design (ICCAD)*, pp. 1–9, 2020.
[26] S. Nikova *et al.*, "Threshold Implementations against Side-Channel Attacks and Glitches," in *Int'l Conference on Information and Communications Security*, 2006, pp. 529–545.
[27] O. Petura *et al.*, "A Survey of AIS-20/31 Compliant TRNG Cores Suitable for FPGA Devices," in *Int'l Conference on Field Programmable Logic and Applications (FPL)*, 2016, pp. 1–10.
[28] P. Barreto *et al.*, "The Khazad Legacy-level Block Cipher," *Primitive submitted to NESSIE*, vol. 97, p. 106, 2000.
[29] J. Daemen, "Cipher and hash function design strategies based on linear and differential cryptanalysis," Ph.D. dissertation, Doctoral Dissertation, March 1995, KU Leuven, 1995.
[30] H. M. Kamali *et al.*, "On Designing Secure and Robust Scan Chain for Protecting Obfuscated Logic," *Great Lakes Symposium on VLSI (GLSVLSI)*, pp. 1–6, 2020.
[31] S. Roshanisefat *et al.*, "DFSSD: Deep Faults and Shallow State Duality, A Provably Strong Obfuscation Solution for Circuits with Restricted Access to Scan Chain," in *VLSI Test Symposium (VTS)*, 2020, pp. 1–6.
[32] W. Diehl *et al.*, "Comparison of Cost of Protection against Differential Power Analysis of Selected Authenticated Ciphers," *Cryptography*, vol. 2, no. 3, p. 26, 2018.
[33] K. Z. Azar *et al.*, "SMT attack: Next Generation Attack on Obfuscated Circuits with Capabilities and Performance beyond the SAT Attacks," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 97–122, 2019.
[34] K. Z. Azar *et al.*, "Threats on logic locking: A decade later," in *Great Lakes Symposium on VLSI (GLSVLSI)*, 2019, pp. 471–476.
[35] P. Subramanyan *et al.*, "Evaluating the Security of Logic Encryption Algorithms," in *Int'l Symposium on Hardware Oriented Security and Trust (HOST)*, 2015, pp. 137–143.