Single-clock-cycle, Multilayer Encryption Algorithm for Single-channel IoT Communications

Shahzad Muzaffar, Owais Talaat Waheed, Zeyar Aung and Ibrahim (Abe) M. Elfadel
Masdar Institute, Khalifa University of Science and Technology, Abu Dhabi, UAE
{smuzaffar,owaheed,zaung,ielfadel}@masdar.ac.ae

***Abstract*—Pulsed-Index Communication (PIC) is a novel technique for single-channel, high-data rate, low-power dynamic signaling that does not require any clock and data recovery. It is fully adapted to the simple yet robust communication needs of Internet of Things (IoT) devices and sensors. However, securing PIC using available conventional symmetric block cipher techniques is not feasible as it significantly degrades PIC attributes of low power, small area, and high data rates. For instance, symmetric stream ciphers such as A5/1 need several clock cycles to encrypt the data, which would reduce the PIC data rate. In this paper, we present a modified A5/1 cipher technique, called MA5/1, that generates a full keystream in one clock cycle, this securing PIC while satisfying all its requirements. Furthermore, using PIC's salient feature of transmitting index pulse streams, we provide an additional layer of packet security that makes it difficult for an attacker to receive and decode the packet before targeting MA5/1. When combined, these two techniques present a two-layer, hard-to-break challenge to an attacker, thus protecting PIC communication in an IoT network. The secure PIC is prototyped and verified in both FPGA and ASIC. In particular, we show that for an ASIC implementation in $65nm$ technology, the low-power operation of PIC is maintained, consuming only $27\mu W$ of power at a clock frequency of $25MHz$.**

I. Introduction

The Internet of Things (IoT) is expected to offer advanced machine-to-machine connectivity of devices, sensors, and systems. Low-end gadgets are expected to be ubiquitous and would constitute the larger percentage of nodes in an IoT network. These low-end nodes include small microcontrollers and processors with very low power consumption, appropriate data rates, small foot print but reliable communication links. It is expected that each of these low-end nodes, sensors or actuators, can be remotely monitored and controlled. Of course only authorised actors are allowed to conduct such monitoring or control, which would require secure authentication and data transfers from one node to another in real time. As in any secure communication link, the goals are to preserve data confidentiality, integrity, and authenticity.

There are many protocols for IoT machine-to-machine communication [6], including our recent work on a low-power, single-channel protocol, named Pulsed-index Communication (PIC) [13], [10], [9], [12], in which the signaling technique does not require any clock and data recovery (CDR). PIC achieves higher date rate, lower power, and smaller footprint as compared to the other available single-channel protocols. More recently, PIC was used as the backbone protocol in a versatile hardware platform for the development and characterization

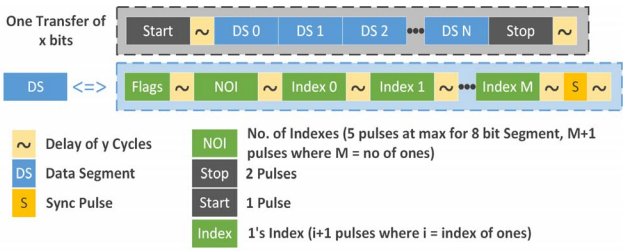

Fig. 1. PIC packet format.

of IoT sensor networks [11]. Such platform can be adapted to any of a number of IoT use cases, including building instrumentation, wearable healthcare, and urban monitoring. The main objective of this work is to show how this platform can be extended to achieve secure communication using a novel lightweight cryptography algorithms that is fully compatible with PIC.

There has been a significant amount of work in the field of light-weight IoT cryptography, including both symmetric and public key techniques. Due to its lower computational requirement, we have selected for PIC symmetric key cryptography. Since the adoption of Advanced Encryption Standard (AES) as a standard in 2002, many light-weight symmetric key block ciphers have been developed. They include CLEFIA, FeW, HIGHT, LBlock, PRESENT, Piccolo, and XTEA [15], [8], [5], [17], [3], [14], [16]. In the case of a single-channel protocol like PIC, block ciphers negatively impact the hardware resources, data rates, and power consumption levels. Symmetric block ciphers for such single-channel cases are not recommended. In contrast, light-weight symmetric key stream ciphers have lower computational demands, and among them, we have selected A5/1 as a candidate cipher to be integrated with PIC. Stream cyphers generate pseudorandom keystreams and encrypt plaintext on a digit-by-digit basis [16]. As will be shown in this paper, such ciphers can be conveniently modified to make them suitable for PIC without compromising the distinctive attributes of the protocol, such as high data rates, low power consumption, and small footprint.

As in any information security task where the protocol is augmented with a cryptographic layer, the main objective is to come up with an encryption and decryption architecture that is simple to implement, with low latency, low power, and small footprint while providing a good degree of defense against

978-1-5090-5569-2/17$31.00

malicious attacks. In this paper, we present a modified version of the A5/1 cipher that we call MA5/1, for Modified A5/1, in which a 16-bit pseudorandom keystream is generated in just one clock cycle based on an initial secret key of 128-bits. The generated keystream is then used to encrypt PIC packets. To further secure PIC, a 4-bit identification number and four other 4-bit keystreams are generated to modify and encrypt different portions of the PIC packet in order to provide an additional level of PIC transmission security and protect the packet at the receiving end. This lightweight, multilayer cryptographic technique provides a hard-to-break challenge to an attacker without compromising PIC performance. The proposed cryptographic solution is implemented in Verilog HDL and verified using the FPGA hardware platform of [11]. The solution is also synthesized using a GLOBALFOUNDRIES $65nm$ for an ASIC implementation that achives lower power consumption than the FPGA one.

This paper is organised as follows. In Section II, we provide the protocol and cryptography background needed to understand and appreciate the contributions of our paper. Section III is devoted to the description of the multiplayer cryptographic system used to secure PIC while Section IV is devoted to the cryptanalysis of such system. The hardware validation of the crypto system is described in Section V. The paper is concluded in VI.

II. Review of PIC and A5/1

In this section, we review the PIC communication protocol and A5/1 stream cypher and provide the background notation and information needed in the rest of the paper.

A. PIC Basics

Pulsed-Index Communication (PIC) [13] is a novel single-channel protocol that does not require any circuitry for CDR. The protocol is based on the novel concept of a pulsed index where instead of transmitting the bits themselves, only the indices of the ON bits in the data word B are transmitted. These indices are encoded as pulse counts. The core of the protocol is to encode these indices so as to *minimize* the number of ON bits, and *move* them to the Least-Significant-Bit (LSB) end of the packet. The encoding process includes a segmentation step where the data is broken into N independent segments of size l bits each (i.e. $N = B/l$). To maximize data rate, PIC uses, on each segment, an encoding combination of bit inversion and/or segment reversion/flipping. This combination is meant to reduce the number of ON bits and decrease their index values so as to lower the number of pulses required to transmit the data bits. To facilitate decoding, flag pulses representing the type of encoding performed (i.e. inversion (1 pulse), reversal (2 pulses), both (3 pulses), or none (4 pulses)) are added to each segment. All the pieces of information including flags, number of indices and the indices themselves are transmitted in the form of pulse streams. The pulse is characterised by its width which is the number of clock cycles during which it remains high. Within a given packet, segment pulse streams are separated by an inter-symbol delay (α) as shown in Fig. 1. The receiver counts the number of pulses for each pulse stream and applies the decoding according to the flags received. The data rate depends on the number of clock cycles required to transmit a packet. Beside the elimination of CDR, the implementation of PIC is area-efficient, low-power and highly tolerant of clocking differences between transmitter and receiver. As empirically found in [13], PIC is dynamic and provides data rates in the range of 3.1-8.5 Mbps with an average of 4.1 Mbps operating at a clock rate of 25 MHz transmitting data words of 16-bits each. In an ASIC implementation on $65nm$ technology, PIC can reduce area by more than 80% and power by more than 70% in comparison with a CDR-based serial bit transfer protocol.

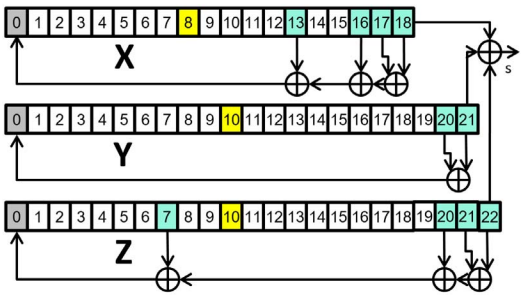

Fig. 2. A5/1 Cipher

For the rest of the paper, we use an implementation of PIC having a word-length of 16 bits. The PIC segmentation breaks the 16-bit data into two equal segments, each of which is encoded separately. Two FLAGs (one for each segment) and two NOIs (one for each segment) are also generated. FLAG represents the segment encoding type and NOI represents the number of ON bits in the segment.

B. A5/1 Cipher Basics

The conventional A5/1 cipher technique [16] is based on a 64-bit secret key which is stored in three registers: 19-bit X $(x_0, x_1, ...x_{18})$, 22-bit Y $(y_0, y_1, ...y_{21})$, and 23-bit Z $(z_0, z_1, ...z_{22})$. The majority function $m = maj(x_8, y_{10}, z_{10})$ decides if the content of each of the registers need to be shifted. The content of a shift register X shifts right (from LSB to MSB) if $m = x_8$, as shown in Fig. 2. A similar shift also occurs to the Y and Z registers after a comparison of m with y_{10} and z_{10}, respectively. The LSB of each register is then replaced with the value, $t_{register}$, determined as follows

$$t_x = x_{13} \oplus x_{16} \oplus x_{17} \oplus x_{18} \quad (1)$$

$$t_y = y_{20} \oplus y_{21} \quad (2)$$

$$t_z = z_7 \oplus z_{20} \oplus x_{21} \oplus x_{22} \quad (3)$$

At each iteration of the above process, the final bit, s, is generated as follows

$$s = x_{18} \oplus y_{21} \oplus z_{22} \quad (4)$$

The generated s is then XORed with one data bit, d_i, to get one encrypted ciphertext bit c_i (i.e. $c_i = d_i \oplus s$). The process needs to be sequentially repeated a number of times

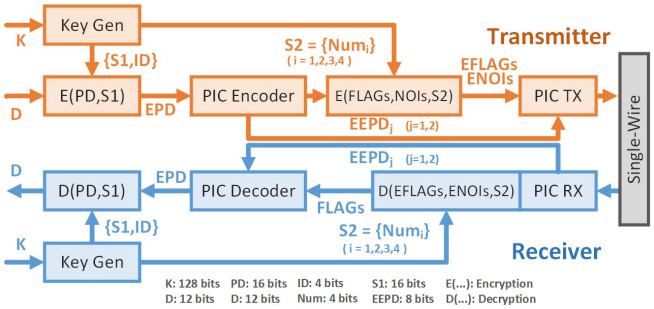

Fig. 3. Secure PIC Transmission Flow

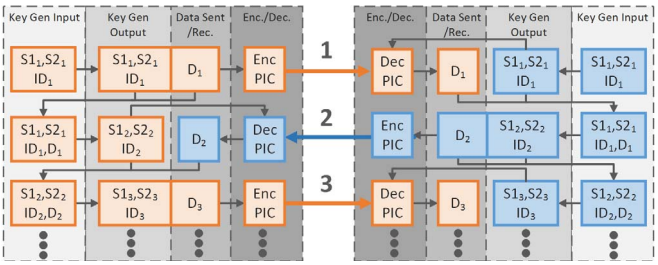

Fig. 4. Example Flow with Key Generator

equal to the number of bits, n, in data word D to generate n number of s bits to produce the cipher text C. A naive implementation would require at least one clock cycle per iteration. The cryptographic clock cycles will of course reduce the PIC data rate of transmission and increase latency at power-on. One possible approach to achieve one clock-cycle operation is hardware parallelism in which the A5/1 block is replicated n times. This approach has several disadvantages. For one thing, it is not possible to use the same 64-bit key for all the copies of A5/1. In fact, n different keys need to be used. For another, hardware parallelism is hungry for both silicon area and resources. In the following section, we introduce a modification of A5/1 that achieves a one clock-cycle operation for a 16-bit keystream generation without A5/1 block duplication. Additionally, we will show how the interplay of PIC and modified A5/1, MA5/1, results in making the PIC transmission more secure than if MA5/1 is used alone in a single-layer cryptographic system.

III. Secure PIC Communication

The complete secure PIC transmission using MA5/1 is comprised of three main components:

1) *Data Encryption*: This component provides confidentiality. A modified version of symmetric stream cipher A5/1 is used to generate key streams.
2) *Packet Encryption*: This component contributes to both confidentiality and integrity by making it difficult for an attacker to successfully receive the packet itself.
3) *Authentication*: This component provides integrity by introducing identification IDs within the packet.

The overall secure PIC transmission flow is shown in Fig. 3, where a transmitter is communicating with a receiver using a single-wire in between. A 4-bit ID for *authentication* purpose is integrated within the 12-bit of input data, D, to form a 16-bit PIC data, PD, to transmit over the single channel. The integrated data is then *encrypted* using a keystream, $S1$. The integrated and encrypted data, EPD, is then encoded by the PIC encoder to maximize the data rate and to generate a PIC packet. As described in the previous section, the PIC encoder generates two 8-bit encoded data segments of EPD, represented by $EEPD_j$ $(j = 1, 2)$ in the flow diagram, two $NOIs$ one for each $EEPD$, and two $FLAGs$ also one for each $EEPD$. The $EEPDs$ are fed directly to the PIC transmitter, but the $NIOs$ and $FLAGs$ are further encrypted using another keystream, $S2$. The $S2$ is a combination of four 4-bit numbers, Num_i $(i = 1, 2, 3, 4)$, each of which is used to encrypt $NOIs$ and $FLAGs$ to generate $ENOIs$ and $EFLAGs$. It must be noted here that $S1$, $S2$, and ID all are generated by the single key generator. The $NOIs$ and $FLAGs$ are then used to transmit $EEPDs$ in the form of a PIC packet over the channel. It is important to know that $NOIs$ and $FLAGs$ only help to transmit $EEPDs$, but are not transmitted themselves. Instead, $ENOIs$ and $EFLAGs$ are transmitted.

At the receiving end, the reverse flow (Fig. 3) is used for reception, decoding, and decryption in order to extract the transmitted data. The receiver first receives the $ENOIs$ and $EFLAGs$ for each of the transmitted segments, recovers the $NOIs$ and $FLAGs$ through decryption, and using these recovered $NOIs$, it logs the upcoming index pulse streams to form $EEPDs$. Afterwards, decoding and decryption are applied to recover PD. PD is then split into D and ID. The $S1$, $S2$, and ID for the receiver are generated by the same key generator as for the transmitter. For both the transmitter and receiver, the set of $S1$, $S2$, and ID is updated at every transaction. The key K and the initial values of Num_i and ID should be same for both the transmitter and receiver. A set of three example transactions is shown in Fig. 4 where all the encoding, decoding, encryption, and decryption processes are indicated by a single box "Enc PIC". The main purpose of the diagram is to show how the inputs and outputs of the key generator change at each transaction to match both ends of the communication link with respect to synchronized keystream generations.

A. Data Encryption

A 16-bit data, PD, is encrypted using a keystream $S1$ that is generated by a modified version of a symmetric key cipher A5/1. Originally, A5/1 generates 1-bit per iteration to XOR it with 1-bit of data. A conceivable efficient hardware implementation of A5/1 is to have each generated bit consume only one clock cycle. If such implementation is used, the encryption of the 16-bit PD in PIC would require at least 16 iterations of A5/1 or 16 clock cycles. This would very negatively impact the PIC data rate. One major contribution of this paper is to propose a modification of A5/1, called $MA5/1$, that will generate one $S1$ of 16 bits in only clock cycle.

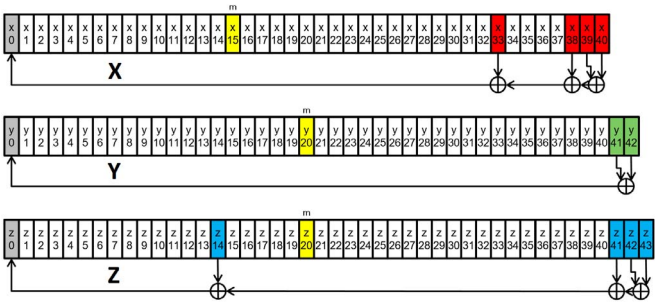

Fig. 5. Modified A5/1 (MA5/1)

TABLE I
16 - BIT KEYSTREAM (S) FUNCTION INDICES

i_S	i_X	i_Y	i_Z	i_S	i_X	i_Y	i_Z
0	5	35	3	8	11	37	32
1	38	27	13	9	32	20	39
2	28	10	23	10	2	11	28
3	8	17	12	11	18	36	34
4	17	39	29	12	19	16	18
5	25	33	24	13	13	25	11
6	12	3	7	14	30	2	20
7	14	23	33	15	10	22	5

1) Properties of MA5/1: The key K of MA5/1 that is used as the initial fill of the shift registers is modified to 128-bits long instead of 64-bits. Similarly, the lengths of shift registers X, Y, and Z are also increased to 41 $(x_0, x_1, ...x_{40})$, 43 $(y_0, y_1, ...y_{42})$, and 44 $(z_0, z_1, ...z_{43})$ bits respectively, as shown in Fig. 5. The majority function (m) and the shift-register step functions $(x_0 = t_x, y_0 = t_y, z_0 = t_z)$ are modified as follows

$$m = maj(\, x_{15}\,,\, y_{20}\,,\, z_{20}\,) \quad (5)$$

$$t_x = x_{33} \oplus x_{38} \oplus x_{39} \oplus x_{40} \quad (6)$$

$$t_y = y_{41} \oplus y_{42} \quad (7)$$

$$t_z = z_{14} \oplus z_{41} \oplus x_{42} \oplus x_{43} \quad (8)$$

The shift register X, Y, and Z step is as in A5/1 except that the operation is executed depending on the comparisons of $m = x_{15}$, $m = y_{20}$, and $m = z_{20}$, respectively. The keystream $(S = s_0, s_1, ...s_{15})$ is then generated as follows

$$S[i_S] = X[i_X] \oplus Y[i_Y] \oplus Z[i_Z] \quad (9)$$

where i_S, i_X, i_Y, and i_Z are the indices given in Table I.

2) Data Dependency of MA5/1 Keystream: In the original A5/1 keystream generation, the keystream will eventually repeat, albeit after a long time. To avoid repetition and to add more randomness in the keystream generation, we have introduced the swapping of shift registers depending on the data transmitted or received during the previous transaction. To achieve this, the 8^{th} bit of the previous data is chosen. If the bit is ON, the shift registers are swapped as follows:

$$X[1:40] = Y[1:40] \quad (10)$$

$$Y[1:40] = Z[1:40] \quad (11)$$

$$Z[1:40] = X[1:40] \quad (12)$$

It must be noted that the 0^{th} bit is not swapped.

B. Packet Encryption

To add another layer of security, the packet encryption exploits the PIC feature of transmitting the index numbers in the form of pulse streams. The number of indices (NOI) for each of the segments of a PIC packet is transmitted prior to the transmission of indices of a following segment. The receiver uses this information to identify the pulse stream of each of the incoming indices. If any of the $NOIs$ is destroyed, the receiver would not be able to receive the packet successfully because the mismatch between the expected number of pulse streams and the count indicated by the destroyed NOI. In this situation, the packet would be discarded. Moreover, the $FLAG$ for each of the segments of a PIC packet is also transmitted prior to the transmission of indices of a respective segment. If any of the $FLAGs$ is destroyed the receiver would not be able to decode the transmitted pulse streams to extract the transmitted data word. This is because a different $FLAG$ would apply a different decoding. By controlling these two features, another layer of security can be added to PIC transmission. This can be done by "destroying" both $NOIs$ and the $FLAGs$ in a way that would confound the attacker but provide the intended receiver with the means to recover the transmitted information. This is bad news for an attacker but not for a receiver. Encrypting $NOIs$ and $FLAGs$ would add an extra shield of packet data protection that would prevent the attacker from even "opening" the packet.

To achieve this extra level of packet security, all the four packet information fields (two $NOIs$ and two $FLAGs$) are encrypted with 4-bit keys which are generated along with the keystream $S1$ in the key generator. These keys, $Num_i\,(i = 1, 2, 3, 4)$, are denoted as keystream $S2$ earlier in the previous subsections and are shown in Fig. 3. Like $S1$, the $S2$ is also updated at each transaction (transmission or reception). To add more randomness, $S2$ also depends on previously transmitted or received data. At the start, each of these keys (Num_i) is initialized with random numbers as per the system's or user's choice. A better option is to distribute these four keys evenly from 0 to 15. The generation of new keys at each transaction follows the functions as follows

$$Num_1 = Num_1 \oplus \{\, PData[5],\, Y[1],\, Z[19],\, X[37]\,\} \quad (13)$$

$$Num_2 = Num_2 \oplus \{\, PData[7],\, Y[7],\, Z[9],\, X[31]\,\} \quad (14)$$

$$Num_3 = Num_3 \oplus \{\, Y[21],\, Z[39],\, X[11],\, PData[5]\,\} \quad (15)$$

$$Num_4 = Num_4 \oplus \{\, Y[13],\, Z[13],\, X[21],\, PData[7]\,\} \quad (16)$$

These keys are then used to generate $ENOI_1$, $FLAG_1$, $ENOI_2$, and $FLAG_2$ as follows

$$ENOI_1 = Num_1 \oplus NOI_1 \quad (17)$$

$$FLAG_1 = Num_2 \oplus FLAG_1 \quad (18)$$

$$ENOI_2 = Num_3 \oplus NOI_2 \quad (19)$$

$$FLAG_2 = Num_4 \oplus FLAG_2 \quad (20)$$

As described previously, the integrated-encrypted-encoded data ($EEPD$) is fed directly to the PIC serializer instead of passing through a second layer of encryption. Only $NOIs$ and $FLAGs$ are encrypted again. A careful look will unveil that $ENOIs$ and $EFLAGs$, which are the encrypted versions of real $NOIs$ and $FLAGs$, are being transmitted. However, the $EEPD$ is transmitted using the real $NOIs$ instead of $ENOIs$. This mismatch will present a major challenge for an attacker that would prevent her from even collecting the full packet successfully, let alone reaching the encrypted data itself. This is because only the receiver can decrypt $ENOIs$ to get real $NOIs$ prior to receiving indices. $ENOIs$, and $EFLAGs$ are transmitted in lieu of the original $NOIs$ and $FLAGs$, but the $EEPD$ is transmitted using the real $NOIs$ instead of $ENOIs$.

C. Authentication

Consider a man-in-the-middle attack where an attacker pretends to be a valid transmitter and tries to send false PIC packets to the receiver. Though it is very difficult for the attacker to generate a completely receivable PIC packet, there is a possibility of success. One attacker's goal is to de-synchronize the receiver key generator so that the receiver starts rejecting packets from a valid transmitter due to $S1$ and $S2$ mismatches. If the attacker is successful in deceiving the receiver, she may just want to interfere with the control of any equipment controlled by the receiver using any invalid data. This is because the receiver has no information on the validity of the transmitter.

Towards transmitter authentication, we include a 4-bit ID, which is generated along with $S1$, $S2$, and Num_i in the key generator, in a 14-bit input data D to produce 16-bit PIC data PD. The ID along with $S1$ is shown in Fig. 4. Like Num_i, the ID is also updated at each transaction (transmission or reception). To add more randomness, ID, like Num_i, is made dependent on the previously transmitted/received data. At the start, the ID is initialized with a random number as per system's or user's choice. The integration of ID in data and the generation of new ID at each transaction are given below

$$PD = \{\, ID[3:2],\, D,\, ID[1:0] \,\} \quad (21)$$

$$ID = ID \oplus \{\, PData[3],\, Y[29],\, Z[21],\, X[9] \,\} \quad (22)$$

Because the PIC packet is broken into two segments prior to the transmission of the encoded segments, ID is integrated in such a way that each of the data segments gets two distinct bits of ID. This is to make sure the ID is recovered only if the complete packet is received and decrypted successfully. At receiver end, after successful decryption of PD, the transmitter ID_T is extracted to compare with the receiver ID_R. In case of failure to authenticate, $ID_T \neq ID_R$, the packet is discarded and none of the $S1$, $S2$, Num_i, and ID is updated.

IV. Cryptanalysis Of Proposed Solution

A considerable amount of literature is available on attacks to break the A5/1 cipher such as guess and clock [1], time-memory trade-off attack [2], and linear equation solve [4]. Time-memory trade-off allows an attacker to reconstruct the key in one second from two minutes of known plaintext at a cost of expensive preprocessing of 2^{48} steps to compute around 300 GB of data. Time complexity to break A5/1 through solving linear equations is $2^{40.16}$. These complexities are for the 64-bit cipher. Our proposed solution not only provides stronger cipher using 128-bit key but also presents the attacker with an even harder challenge through PIC packet protection and transmitter's identification. The modifications to A5/1 and its integration with PIC protocol makes the secure transmission much stronger. Not only does it add confusion to the cipher system but also diffusion by making the settings of the MA5/1 registers dynamically dependent on the plaintext.

To reach the main MA5/1 cipher system, an attacker has first to challenge herself to receive the packet successfully. The plaintext-dependent encrypted information for the number of indices ($ENOI$s) need to be decrypted successfully prior to the reception of indices, otherwise packet failure occurs. Furthermore, successful reception of an encrypted encoding type ($EFLAGs$) depends on $ENOIs$ itself, which provides additional protection to the decoding process. If an attacker is successful in guessing both of these then the transmitter identification (ID) would be the next barrier. Recall that all of these three packet fields are updated at each packet transmission along with the updated keystream. Moreover, like the keystream, these keys are plaintext-dependent, which makes it difficult for an attacker to send any invalid data to destroy the synchronization or control of the device. The probability that an attacker could guess a packet that is acceptable for the receiver is

$$P_{att.} = \frac{1}{2^{12} \times 2^4 \times 2^4 \times 2^4 \times 2^4 \times 2^4} \approx 0.23 \times 10^{-9} \quad (23)$$

This is because an attacker has to guess two 4-bit $NOIs$, two 4-bit $FLAGs$, one 4-bit ID, and a 12-bit D to form a valid PIC packet.

Another issue that could arise is when we have constant data as, for instance, when the sensor node is powered off and the digital value is all zeros. In this case, MA5/1 reduces to A5/1 and can therefore be broken as long as the A5/1 key is guessed correctly. Again, to reach this point, successful packet reception and its decoding are necessary. To further mitigate this issue, we recommend to use the MA5/1 register contents after N cycles by discarding the initial 256-bits of s. In case an attacker wants to store a transmitted packet and then applies exhaustive search to find the key, she has two main challenges. First, if we suppose that she is able to recover the key, she must know the previously transmitted data as the keystream and register contents are data dependent. In most cases, previously transmitted data is not known to her. Second, even in case such data is known to the attacker, she must adopt a brute-force exhaustive search method with insurmountable complexity given by

$$W_{exh} = \frac{2^{128} \times 2^{20}}{2} = 2^{147} \quad (24)$$

TABLE II
SYNTHESIS RESULTS

is	**Power** (μW)	**Area** ($gate\ count$)
Secure PIC (Excluding PIC)	27	≈2780
PIC	26.6	2356
Total	53.6	≈5136

V. EXPERIMENTAL VERIFICATION

An experimental setup comprised of two IoT nodes communicating using PIC is used to verify the operation of the proposed secure communication solution. Each node is composed of a keystream generator, a controller, and a PIC protocol module. Using the keystreams generated by the key generator and the PIC transmission and reception modules, the controller module is responsible for all the communication, encryption, decryption, and authentication flows. The full experimental setup is implemented in Verilog on the Xilinx Virtex-7 FPGA platform. The setup transmits data securely from one node to the other, and the other node sends received data back to the first node to complete a full transmission link. Received and transmitted data are then compared to verify two-way communication. Rigorous simulation and hardware experimentation are performed for functional testing and verification of the implemented system. The experiments confirm that the secure PIC transmission works flawlessly. Along with the FPGA prototype, we have also synthesized the system using a Synopsis logic-synthesis flow and a GLOBALFOUNDRIES $65nm$ process. We found out that the system shown in Fig. 3 (excluding PIC) maintains the low-power operation of PIC consuming only $27\mu W$ with a gate count of ≈2780 at a clock frequency of $25MHz$. The synthesis results are shown in Table II.

VI. CONCLUSIONS

The proposed single-channel, secure communication system exploits the unique features of a particular single-channel protocol, PIC, and adds multilayer security to the transmission with low impact on PIC performance while presenting a set of hard-to-solve challenges to an attacker. This secure systems is based on a modified version of A5/1, MA5/1, that uses a total key length of 148-bit, out of which 128-bits are used for the MA5/1 keystream generation and 20-bits are used to make PIC more secure through an additional layer of packet encryption. The proposed solution features both confusion and diffusion defences and inherently provides both data confidentiality and sender authentication. The proposed secure system achieves high data rates, low power consumption, and small footprint, all in line with the original PIC attributes. Conceivably, MA5/1 may be used with other single-channel protocols albeit without the additional crypto layer that is tied up to the PIC packet specification. The secure systems has been implemented in both FPGA and ASIC (GLOBALFOUNDRIES 65nm) platforms.

Related topics for future work include comparing the proposed MA5/1 cypher with some of the other famous light-weight stream cipher techniques such as GRAIN, MICKEY, and Trivium [7]. Moreover, a more complete cryptanalysis of MA5/1, as deployed over PIC, will need to be performed to further clarify its advantages in comparison with other light-weight cryptographic techniques.

VII. ACKNOWLEDGMENT

This work has been supported by the Semiconductor Research Corporation (SRC) under the Abu Dhabi-SRC Center of Excellence on Energy-Efficient Electronic Systems (ACE4S), Contract 2013 HJ2440, with funding from the Mubadala Development Company, Abu Dhabi, UAE.

REFERENCES

[1] Ross Anderso. A5 (was: Hacking digital phones). *Newsgroup: uk.telecom*.
[2] Alex Biryukov, Adi Shamir, and David Wagner. Real time cryptanalysis of a5/1 on a pc. *Fast Software Encryption Workshop 2000*, April 2000.
[3] A. Bogdanov, L.R. Knudsen, G. Leander, C. Paar, A. Poschmann, M.J. Robshaw, Y. Seurin, and C. Vikkelsoe. Present: An ultra-lightweight block cipher. *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 450–466, 2007.
[4] Jovan Dj. Golic. Cryptanalysis of alleged a5 stream cipher. *International Conference on the Theory and Application of Cryptographic Techniques*, pages 239–255, May 1997.
[5] D. Hong, J. Sung, S. Hong, J. Lim, S. Lee, B.S. Koo, C. Lee, D. Chang, J. Lee, K. Jeong, and H. Kim. Hight: A new block cipher suitable for low-resource device. *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 46–59, 2006.
[6] IoT Standards and Protocols, Postscapes.
[7] Masanobu Katagi and Shiho Moria. Lightweight cryptography for the internet of things. *Sony Corporation*, 2012.
[8] M. Kumar, S.K. Pal, and A. Panigrahi. Few: A lightweight block cipher. *IACR Cryptology ePrint Archive*, page 326, 2014.
[9] S. Muzaffar and I. M. Elfadel. Power management of pulsed-index communication protocols. *IEEE International Conference on Computer Design (ICCD)*, pages 375–378, October 2015.
[10] S. Muzaffar and I. M. Elfadel. Timing and robustness analysis of pulsed-index protocols for single-channel IoT communications. *IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*, pages 225–230, October 2015.
[11] S. Muzaffar and I. M. Elfadel. A versatile hardware platform for the development and characterization of iot sensor networks. *IEEE Midwest Symposium on Circuits and Systems (MWSCAS)*, pages 1–4, October 2016.
[12] S. Muzaffar, Numan Saeed, and I. M. Elfadel. Automatic protocol configuration in single-channel low-power dynamic signaling for IoT devices. *IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*, pages 1–6, September 2016.
[13] Shahzad Muzaffar, Ayman Shabra, Jerald Yoo, and Ibrahim (Abe) M. Elfadel. A pulsed-index technique for single-channel, lowpower, dynamic signaling. *Design, Automation and Test In Europe(DATE)*, pages 1485–1490, March 2015.
[14] K. Shibutani, T. Isobe, H. Hiwatari, A. Mitsuda, T. Akishita, and T. Shirai. Piccolo: an ultra-lightweight blockcipher. *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 342–357, 2011.
[15] T. Shirai and et al. The 128-bit blockcipher clefia. *International Workshop on Fast Software Encryption*, 2007.
[16] Mark Stamp. *Information Security: Principles and Practices, 2nd Edition*. Wiley, USA, 2011.
[17] W. Wu and L. Zhang. Lblock: a lightweight block cipher. *International Conference on Applied Cryptography and Network Security*, pages 327–344, 2011.