Integral Cryptanalysis of the BSPN Block Cipher

Howard Heys
Department of Electrical and Computer Engineering
Memorial University
hheys@mun.ca

Abstract— **In this paper, we investigate the application of integral cryptanalysis to the Byte-oriented Substitution Permutation Network (BSPN) block cipher. The BSPN block cipher has been shown to be an efficient block cipher structure, particularly for environments using 8-bit microcontrollers. In our analysis, we are able to show that integral cryptanalysis has limited success when applied to BSPN. A first order attack, based on a deterministic integral, is only applicable to structures with 3 or fewer rounds, while higher order attacks and attacks using a probabilistic integral were found to be only applicable to structures with 4 or less rounds. Since a typical BSPN block cipher is recommended to have 8 or more rounds, it is expected that the BSPN structure is resistant to integral cryptanalysis.**

Index Terms— **cryptography, block ciphers, cryptanalysis**

I. Introduction

The Byte-oriented Substitution Permutation Network (BSPN) structure for use in block cipher design was first proposed in [1][1]. BSPN is not a specific cipher proposal with defined components, but rather a structure for block cipher architectures belonging to the class referred to as Substitution Permutation Networks (SPNs). The BSPN block cipher structure has been analyzed and proposed for applications using 8-bit microcontrollers. In [1], it is shown that a 64-bit BSPN can be resistant to differential cryptanalysis [2] and linear cryptanalysis [3] for 8 rounds of operations with a suitable selection for the S-boxes. In [4], a 64-bit BSPN cipher is analyzed for use in sensor nodes of wireless sensor networks and is shown to be more energy efficient than implementations of the Advanced Encryption Standard (AES) [5]. The efficiency of BSPN is also discussed in Section II.

In general, SPNs are constructed with the datapath consisting of rounds of key mixing, nonlinear byte substitution (using an S-box), and a linear transformation[2]. The principles of SPN architectures are rooted in Shannon's concept of a product cipher, constructed to ensure confusion and diffusion [6]. SPNs can be efficiently implemented in both hardware and software and can be constructed to be secure.

Figure 1 illustrates the BSPN architecture. As proposed in [1], BSPN is defined as an involution cipher. This property allows the use of the same components for both encryption and decryption, thereby facilitating efficient implementations based on the re-use of components or memory when both encryption and decryption operations are required. In order to achieve involution, the use of involutory S-boxes and a linear transformation that is an involution are required. For BSPN, as defined in [1], no particular S-box is specified. However, the S-box is assumed to be 8×8 (where one input byte is replaced with an output byte) and must satisfy specific properties, such as involution, small linear bias, and small maximum differential.

The BSPN is most notably characterized by its linear transformation, for which a specific function is defined. Let the block size of the cipher be N bytes or $8N$ bits with $U_i = [U_{i1}, U_{i2}, ..., U_{i8}]$ and $V_j = [V_{j1}, V_{j2}, ..., V_{j8}]$ representing the i-th input and j-th output bytes of the linear transformation, respectively, where $U_{il} \in \{0,1\}$, $V_{jl} \in \{0,1\}$, $1 \leq l \leq 8$, and $1 \leq i, j \leq N$. The linear transformation is defined as the summation of $N-1$ bytes as in:

$$V_j = \bigoplus_{i=1, i \neq j}^{N} U_i \tag{1}$$

where the summation of two bytes, $X = [X_1, X_2, ..., X_8]$, $X_i \in \{0,1\}$, and $Y = [Y_1, Y_2, ..., Y_8]$, $Y_i \in \{0,1\}$, is defined as the bitwise XOR of the bytes, that is, $X \oplus Y = [X_1 \oplus Y_1, X_2 \oplus Y_2, ..., X_8 \oplus Y_8]$.

The key scheduling process is not defined for BSPN, but in [1] a process to generate the round keys using the full un-keyed cipher is suggested. A round key of a size equal to the block size is mixed into the cipher data by using a bitwise XOR. Note that the last round of an R round cipher differs from the first $R-1$ rounds in that the linear transformation is replaced by a last layer of round key mixing, resulting in the requirement of $R+1$ round keys. This is typical in SPN structures and is required in order to ensure that the decryption process (which is equivalent to travelling backwards through Figure 1) is similar in structure to encryption.

Although the BSPN structure does seem resistant to classical block cipher attacks such as differential and linear cryptanalysis, the applicability of integral cryptanalysis [7] has not been considered for the BSPN structure. However, it is expected that modern block cipher proposals consider resistance to the integral attack, particularly when the structure might lead to susceptibility. Hence, in order to confirm BSPN's resistance to integral attacks, in this paper, we explore

[1]Although the BSPN structure was proposed in [1], the first use of the label "BSPN" appears in [4].

[2]Here, we use the term SPN to apply to ciphers with a general linear transformation for the "Permutation" layer, although, strictly, the first SPN's proposed a permutation of bit ordering as the linear transformation. Our broad definition of an SPN allows AES to also be included in the class of SPNs.

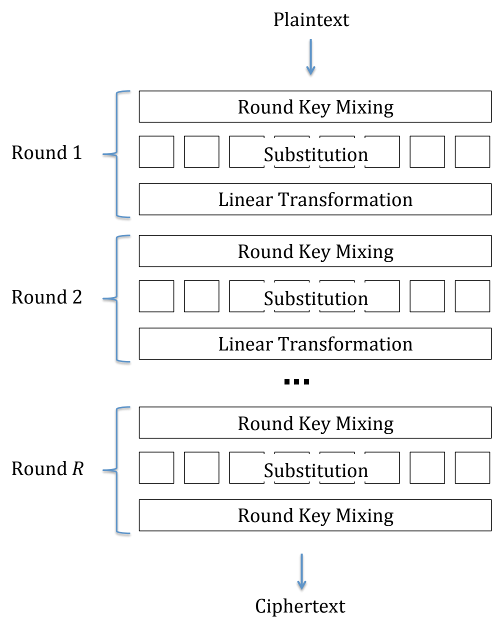

Fig. 1. BSPN Structure

the application of integral cryptanalysis and determine the resulting strength of BSPN in relation to the attack.

II. Efficiency of BSPN Implementations

Before we consider the application of integral cryptanalysis to the BSPN cipher structure, we first discuss the value of the cipher in relation to the efficiency of its underlying operations. Although the 64-bit version of the cipher is shown in [4] to be efficient in the context of an 8-bit microcontroller implementation targeted to wireless sensor network nodes, this comparison is made relative to AES which has a 128-bit block. Although it can be expected that in many embedded applications, a 64-bit block is sufficient, even preferred for efficiency, it would be interesting to consider the comparison of a BSPN cipher similarly sized to AES. Hence, consider a BSPN based on (1) with $N = 16$. In this case, as with AES, the S-box layer of BSPN requires 16 memory lookups to achieve the necessary nonlinear substitutions for an 8-bit microcontroller implementation. In both ciphers, key mixing consists of XORing a block of round key bits with cipher data. The appreciable difference then between the ciphers is the composition of the linear transformation.

Consider first the encryption process. For AES, the linear transformation consists of the *ShiftRows* and *MixColumns* operations, with the *MixColumns* operation requiring significant computation. It is well known that 8-bit microcontroller implementations, in executing *MixColumns*, can make use of the *xtime* operation [5], which performs multiplication by 2 in the GF(2^8) field selected for AES. The *xtime* operation does this using a shift by 1 bit, followed by a conditional XOR of the data byte with a fixed byte defined by the field. In an implementation of AES, 24 XORs of bytes and 4 *xtime* operations are required to execute *MixColumns* on 32 bits. For the full 128 bit block, this translates to the requirement of 96 XORs plus 16 *xtime* operations. In comparison, for the 128-bit BSPN, the linear transformation requires 31 XORs to execute. This is achieved by (a) first computing the XOR of all 16 bytes (using 15 XORs), followed by (b) 16 XORs, one XOR of the resulting byte of (a) with each byte of the block. Hence, on a per-round basis, considering only the encryption operation, the linear transformation of BSPN requires no *xtime* operations and less than 1/3 of the XOR operations required to implement the AES linear transformation.

Considering now decryption, the BSPN has even greater advantage in efficiency. It is well known that the direct implementation of *InvMixColumns* operation of AES is much less efficient than the *MixColumns* operation used in encryption. Conversely, the inverse linear transformation in the BSPN cipher is identical to the linear transformation used in encryption. (This follows from the fact that BSPN is designed to be a cipher with involutory components.) Hence, the linear transformation operation of BSPN is substantially more efficient than AES for decryption. Further, since the S-boxes used for decryption in BSPN are identical to the S-boxes used in encryption (again, following from the involution nature of the cipher), then, while AES needs to store in memory both 256 bytes for the encryption S-box and 256 bytes for the decryption S-box, BSPN only requires the storage of one 256 byte table for the S-box.

It can also be shown that BSPN can be very compactly and efficiently implemented in target hardware environments such as FPGAs and ASICs, as well as, microcontrollers with datapath widths larger than 8 bits. However, a full discussion of this is outside the scope of this paper.

III. Integral Cryptanalysis

Integral cryptanalysis was first presented as an attack on the Square block cipher (a predecessor of AES) and was initially referred to as the Square attack [8]. Subsequently, similar attacks have been applied to various block cipher structures, under the names of saturation attack [9], structural cryptanalysis [10], integral cryptanalysis [7], and multiset cryptanalysis [11]. In our work, we use the terminology associated with integral cryptanalysis. We first describe a first order integral attack which makes use of a deterministic integral.

The basic concept of integral cryptanalysis as applied to a 64-bit block cipher is shown in Figure 2. Integral cryptanalysis is a chosen plaintext attack, requiring the cryptanalyst to obtain sets of ciphertexts corresponding to plaintexts with a specific relationship. This relationship (as shown in Figure 2) is such that all bytes of the plaintext block, but one, are held constant at arbitrary values (indicated by "C" in the plaintext block), while the remaining byte is cycled through all 256 possible values from all zeroes to all ones (indicated by "A" in the plaintext block). At the output of round $R-1$ (where it is assumed that the cipher has R rounds), for ciphers susceptible

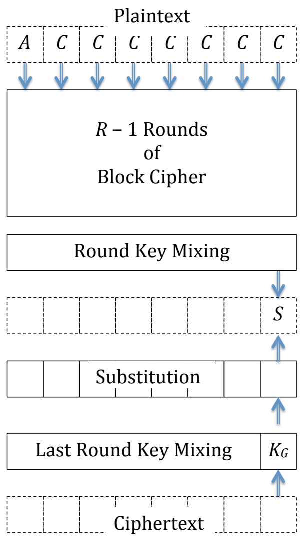

Fig. 2. Implementation of Integral Cryptanalysis on 64-bit Block Cipher

to the attack, at least one byte can be observed where the integral, I, is certain to be zero, that is, $I = [0, 0, ..., 0]$. An integral is defined as an XOR sum as follows:

$$I = \bigoplus_{i=1}^{256} X_i \tag{2}$$

with X_i representing the byte after the round key mixing at the input to the S-box layer in round R. In Figure 2, a byte with integral I of zero is indicated by "S".

In general, for a strong cipher, the integral is to be random and therefore the expected integral would be one of 256 possible values with equal probability. However, since, for a weak cipher, the integral is zero with a probability 1, when guessing round key bits in the last round, the integral can be used to distinguish the correct key guess from the incorrect guesses. This is done as follows: the cryptanalyst guesses all values of the last round key for the byte associated with the integral byte (indicated by K_G in Figure 2). For each guess, all ciphertext values are decrypted one round (that is, through the last round key and last layer of S-boxes) to form the byte value associated with the integral byte. The integral is then calculated by summing over all bytes generated for the set of ciphertexts. For the correct key guess, the integral will be zero with certainty, while for each incorrect guess, the integral will not be zero with a probability 255/256 (assuming that the decryption using the wrong key results in a random integral). So an integral of zero corresponds to a set of key guesses containing the correct key and a small number of spurious keys (possibly none). To eliminate spurious keys, the attacker can use another set of plaintexts to generate a new integral for testing. The probability of the same spurious key being in the set of keys with a zero integral for both cases is very small. Hence, if a key appears in both sets, we can assume that it is the correct key.

The complexity of this basic attack is very low. Assuming no spurious keys in the original set, we would only require 256 chosen plaintexts and associated ciphertexts to attack the cipher, while requiring an analysis for the integrals over 256 ciphertexts for each of 256 guesses of the round key byte, K_G, for the final round. Hence, 2^{16} operations (where an operation is a partial decryption) are required for the attack and 2^8 chosen plaintexts and associated ciphertexts are utilized.

IV. Application of First Order Integral Cryptanalysis to BSPN

In order to apply the first order attack to BSPN, we must consider the maximum number of rounds for which at least one output byte is guaranteed to have an integral of zero. We focus our discussion on the BSPN structure with a 64-bit block size; however, the same results and conclusions apply to larger block sizes. For BSPN, after 2 rounds (plus the key mixing before the 3rd round S-boxes), the integrals of all datapath bytes are zero. For 3 rounds, the integrals of all bytes are zero with a probability substantially less than one. Hence, the first order integral attack is applicable to a 3 round version of BSPN (using a 2 round integral), but not applicable to versions of the cipher with 4 or more rounds (requiring integrals of 3 or more rounds).

We now show the correctness of these results by first considering some basic axioms. Recall that an integral is formed for a byte in the cipher datapath by summing over all byte values in the set corresponding to the set of 256 plaintexts used in the attack. Let C represent the set of integrals corresponding to all bytes in the integral being constant, A represent the set of integrals for which the byte cycles through all possible values (although not in any specific order) over the set defined by the plaintext set, and S represent the set of integrals which evaluate to zero over the set defined by the plaintext set. Further, let X represent the byte over which an integral is taken, while f and g represent the S-box and linear transformation functions, respectively. Note that f maps an 8-bit input to an 8-bit output and g takes $N = 8$ bytes as input to produce an output byte as per (1).

The axioms of Table I are trivially true and can be easily proven (see [10]). Axioms 1 and 2 are trivial properties of integrals. Axioms 3, 4, and 5 represent the operation of the round key mixing and, in cases involving constant bytes, the linear transformation. The effect of the S-box operation on an integral is given in Axioms 6, 7, and 8 and Axiom 9 is a simple statement that the linear transformation does not affect the result of the integration, as the integration is itself a linear operation.

Consider now subsequent rounds of BSPN. At the input, using the plaintext set with a byte from the set A on the left (which we shall refer to as byte 1) and the remaining bytes (from left to right, bytes 2 to 8) as constant bytes, clearly all input bytes have a zero integral. The transformations of the integral properties through the subsequent operations of the

TABLE I

AXIOMS FOR INTEGRAL CRYPTANALYSIS

Axiom Num.	Axiom	Interpretation
1	$X \in C \implies X \in S$	Any byte that is constant has an integral of zero.
2	$X \in A \implies X \in S$	Any byte that cycles through all values has an integral of zero.
3	$X \in C, k \in C \implies X \oplus k \in C$	Adding a constant to a constant results in a constant.
4	$X \in A, k \in C \implies X \oplus k \in A$	Adding a constant to a byte that cycles through all values results in a byte that cycles through all values.
5	$X \in S, k \in C \implies X \oplus k \in S$	Adding a constant to a byte that has a zero integral results in a byte that has a zero integral.
6	$X \in C \implies f(X) \in C$	The application of the S-box to a constant byte results in a constant byte.
7	$X \in A \implies f(X) \in A$	The application of the S-box to a byte which cycles through all values results in a byte that cycles through all values.
8	$X \in S \not\Longrightarrow f(X) \in S$	The application of the S-box to a byte which has a zero integral. does not necessarily result in a byte with a zero integral.
9	$X_1 \in S, ..., X_N \in S,$ $\implies g(X_1, ..., X_N) \in S$	The application of the linear transform to bytes that have integrals of zero, results in a byte that has an integral of zero.

cipher are illustrated in Table II, which lists the integral of the output of the operation, as well as the output properties, based on the inputs to the given operation. (Bytes for which the integral does not belong to the sets with the properties described above are indicated by a dash in Table II.) The output properties and the integrals are determined by making use of the indicated axioms found in Table I. For example, for the round 1 linear transformation, each addition is either the sum of all constant bytes (which can be taken as pairwise sums using Axiom 3) or the sum of 6 constant bytes and one byte from set A (which can be considered as added to the constant byte which is the sum of 6 constant bytes); the integral of each byte from set A or set C must be zero and therefore all bytes also belong to set S as per Axioms 1 and 2.

From Table II, we can see that the integral at the output of the second round (and, consequently, after the key mixing of the 3rd round) is zero. Hence, as described in Section III, the cryptanalyst can mount a successful attack on a cipher of 3 rounds. After the S-box of the 3rd round, the integral of all bytes can no longer be guaranteed to be zero and, hence, the attack cannot be applied to ciphers comprised of 4 or more rounds.

V. HIGHER ORDER INTEGRAL CRYPTANALYSIS OF BSPN

Higher order integral cryptanalysis was introduced in [7] and applied to AES. Due to the linear transformation structure (i.e., *ShiftRows* and *MixColumns*) of AES, it is shown that a fourth order integral attack is applicable up to 6 rounds of AES. This results in an improvement in complexity over the 6 round attack possible with first order integral cryptanalysis.

For a 64-bit BSPN, in applying a second order attack, using a chosen plaintext approach, 6 of the 8 input bytes are held at an arbitrary fixed value, while the remaining two bytes are cycled through all possible 2^{16} values. At the output of the $R-1$ rounds, the integral for a byte is considered as the sum of 2^{16} values in response to the set of plaintext values. The resulting higher order integral, I, can be represented as

$$I = \bigoplus_{i=1}^{256^T} X_i \tag{3}$$

with X_i representing the byte at the output of round $R-1$ corresponding to the i-th plaintext input and for the second order attack $T = 2$. Of course, there are 2^8 possible outcomes for the integral of (3), which for a strong cipher, would be expected to be equally likely. In applying integral cryptanalysis to a weak cipher, we consider the case where a zero integral has a probability 1 of occurring. In this case, we can apply the same principles as for the first order attack to obtain a byte of the last round key. More generally, for an attack of order T, T bytes of plaintext are cycled through all possible values, while the remaining $8-T$ bytes are kept constant. Again, cases where integrals of zero are known to occur can be exploited to find bits of the last round key.

We have experimentally applied the basic approach of higher order integral cryptanalysis to a 64-bit BSPN (based exclusively on the AES S-box) for $T = 2$, 3, and 4 using simulated data, and have found that integrals equal to zero occur with probability 1 at the output of round 3 for all values of T, implying that it is possible to apply a higher order integral attack to a 4 round implementation of BSPN. Hence, for the second order attack (which is the most efficient), it will take 2^{16} chosen plaintexts, with a total of 2^{24} partial decryptions to determine a key byte from the last round of the cipher.

VI. PROBABILISTIC INTEGRAL CRYPTANALYSIS

The previous description of integral cryptanalysis is deterministic in that it relies on the certainty that the integral of the targeted byte at the output of the round $R-1$ has an integral of zero. This knowledge is then used to distinguish a correct (for which the certainty exists) and an incorrect key byte (for which the probability of the integral being zero is expected to be 1/256). In this section, we explore making use of integrals that are not zero with probability 1, but with a

TABLE II

PROVABLE FIRST ORDER INTEGRAL PROPERTIES OF BSPN

Operation	Input Bytes		Output Bytes		Axioms
	Properties	Integrals	Properties	Integrals	
Plaintext	$ACCCCCCC$	$SSSSSSSS$			1, 2
Round 1 Key Mixing	$ACCCCCCC$	$SSSSSSSS$	$ACCCCCCC$	$SSSSSSSS$	1,2,3,4
Round 1 S-box	$ACCCCCCC$	$SSSSSSSS$	$ACCCCCCC$	$SSSSSSSS$	1,2,6,7
Round 1 Linear Transform	$ACCCCCCC$	$SSSSSSSS$	$CAAAAAAA$	$SSSSSSSS$	1,2,3,4
Round 2 Key Mixing	$CAAAAAAA$	$SSSSSSSS$	$CAAAAAAA$	$SSSSSSSS$	1,2,3,4
Round 2 S-box	$CAAAAAAA$	$SSSSSSSS$	$CAAAAAAA$	$SSSSSSSS$	1,2,6,7
Round 2 Linear Transform	$CAAAAAAA$	$SSSSSSSS$	- - - - - - - -	$SSSSSSSS$	1,2,9
Round 3 Key Mixing	- - - - - - - -	$SSSSSSSS$	- - - - - - - -	$SSSSSSSS$	5
Round 3 S-box	- - - - - - - -	- - - - - - - -	- - - - - - - -	- - - - - - - -	8

probability that is substantially different than the probability that a random byte has an integral of zero, which is assumed to be 1/256.

Consider the first order attack. Examining the integrals at the output of the third round S-boxes, we have found some bytes have a probability substantially different than 1/256, although less than 1. Experimental probabilities for 10^7 randomly generated integrals for the outputs of the round 3 S-boxes ranged from .0050143 to .0050429 (except the leftmost S-box, S-box 1, for which the value was .0039197). The expected random probability is 1/256 = .00390625 and, while this probability is very close to the observed probability for S-box 1, the remaining S-boxes have probabilities of about .005, which is clearly distinguishable and can be shown to have a difference of statistical significance from the random probability. If a cryptanalyst could use the information from one of these 7 S-boxes, it would be conceivable to distinguish the case for the correct key from the incorrect keys. However, considering Figure 1, for a 4 round cipher (that is, $R = 4$), in order to examine the information at the output of the round 3 S-boxes, it is necessary to decrypt (for each guess of the last round key byte) through the S-box layer plus the linear transformation layer of round 3. This makes it not possible to contain the guessed round key to only one byte, since many bytes influence one output byte of the inverse linear transformation.

In order to get around this problem, the cryptanalyst can make use of the fact that the XOR sum of two bytes at the output of the linear transformation is simply the sum of the corresponding two input bytes. That is,

$$V_i \oplus V_j = U_i \oplus U_j \tag{4}$$

where $\{V_i,\ V_j\}$ and $\{U_i,\ U_j\}$ represent bytes of the linear transformation output and input, respectively. This can be easily proven by considering (1).

Table III lists the experimental probabilities from 10^7 randomly generated integrals from the XOR of pairs of bytes at the output the round 3 linear transformation, which according to (4), is also equal to the sum of the corresponding two bytes at the output of the round 3 S-boxes. Since the integrals of the bytes at the output of these S-boxes is not expected to be random, it is reasonable to expect that the XOR of two such bytes will also not be random. In fact, as can be determined from Table III, the integral of the sum of the two bytes is expected to be zero with a probability of about .00414, except when byte 1 is involved, where the integral of the XOR sum appears to be zero with the probability of a random integral.

A probabilistic attack on the 4 round cipher can proceed as follows. Consider the use of n sets of plaintexts for the first order attack, that is, a total of $256n$ plaintexts to generate n integrals by taking the pairwise XOR at the output of the round 3 linear transformation. For each ciphertext, two bytes of the last round key are guessed and decryption is undertaken to derive the output of the round key mixing at the input to the round 4 S-boxes (essentially equivalent to the output of the round 3 linear transformation in terms of the integral). For a correct guess of the 16 bits of the target round key bytes, at the output of the 3rd round, the probability that the integral will be zero will be about .00414, or, in other words, about $.00414n$ values of the integrals will be zero, whereas for an incorrect key guess, about $n/256$ integrals will evaluate to zero.

Out of a total of n integrals, the probability that m integrals evaluate to zero is derived in both cases by the binomial distribution:

$$P_S(m) = \binom{n}{m} \times \Lambda^m \times (1-\Lambda)^{n-m}, \tag{5}$$

where for the case of the correct key, $\Lambda = .00414$ and for the incorrect key, $\Lambda = 1/256$.

Let μ_0 (μ_1) represent the expected number of zero integrals for the incorrect (correct) key and σ_0^2 (σ_1^2) be the variance of m. For the binomial distribution, we have

$$\begin{aligned} &\mu_0 = (1/256) \cdot n \\ &\sigma_0^2 = n \times (1/256) \times (1 - 1/256) = .003891 \cdot n. \end{aligned} \tag{6}$$

and

$$\begin{aligned} &\mu_1 = .00414 \cdot n \\ &\sigma_1^2 = n \times .00414 \times (1 - .00414) = .004123 \cdot n. \end{aligned} \tag{7}$$

In attacking the cipher, the cryptanalyst can set a threshold γ so that, if, for the n integral values calculated for the guessed target partial round key, m are zero and $m > \gamma$, then the key can assumed to be correct (since more zero integrals are expected for the correct key) and for $m < \gamma$, the key can be assumed to be incorrect. Using this process, we can approximate the binomial distributions as Guassian with the

TABLE III

PROBABILITY OF ZERO INTEGRAL FOR THE SUM OF TWO BYTES AFTER ROUND 3 LINEAR TRANSFORMATION

Byte Numbers	1+2	1+3	1+4	1+5	1+6	1+7	1+8
Probability	.0039022	.0039102	.0039014	.0039266	.0039301	.0039361	.0039489
Byte Numbers	2+3	2+4	2+5	2+6	2+7	2+8	3+4
Probability	.0041021	.0041446	.0041475	.0041379	.0041736	.0041592	.0041153
Byte Numbers	3+5	3+6	3+7	3+8	4+5	4+6	4+7
Probability	.0041641	.0041098	.0041427	.0041686	.0041446	.0041731	.0041600
Byte Numbers	4+8	5+6	5+7	5+8	6+7	6+8	7+8
Probability	.0041423	.0041635	.0041491	.0041513	.0041424	.0041355	.0041552

indicated means and variances to estimate the likelihood of the attack success. For example, let $\gamma > \mu_0 + 3 \times \sigma_0$ and $\gamma < \mu_1 - 2 \times \sigma_1$. As a result, we can estimate that the correct key is recognized with probability of at least 97.72%, as given by the area of the Guassian distribution above 2 standard deviations below the mean. An incorrect key is improperly assumed to be correct with a probability of less than .13% (implying all 255 incorrect keys have a probability $<<$ 1 of having one or more appear correct), as determined by the tail of the Gaussian distribution above 3 standard deviations above the mean. Making use of (6) and (7), it can be shown that these conditions for γ are met when $n > 1.82 \times 10^6$. Hence, such an attack requires $1.82 \times 10^6 \times 256 \approx 2^{29}$ chosen plaintexts to implement, where each ciphertext requires the analysis of 2^{16} key guesses in order to determine the targeted two bytes of the last round key of the cipher. Finding 7 of the 8 bytes of the last round key can be derived in this way (since byte 1 does not provide useful information) and then this knowledge can typically be easily extended to find the 8th byte of the last round key and subsequently the remaining round keys.

The two probabilities associated with a successful attack can be modified by moving the location of the decision threshold γ. Moving it closer to μ_1, for example, will decrease the probability that the correct key is recognized, but decrease the probability that an incorrect key will be mistakenly determined to be correct. Also, increasing the value of n can be used to increase the probability of the attack success.

The probabilistic first order integral attack takes about $2^{29} \times 2^8 = 2^{37}$ chosen plaintexts of data and has a complexity of $2^{37} \times 2^{16} = 2^{53}$ partial decyption operations to find two key bytes in the last round of a 4 round BSPN. In comparison, a second order deterministic attack on a 4 round BSPN only requires the analysis of one second order integral resulting from 2^{16} chosen plaintexts, with a complexity of 2^{24} partial decryption operations. For second (and higher) order attacks, we have also experimentally examined the probabilities of zero integrals for rounds exceeding the number of rounds required in the deterministic attack and have not observed meaningful statistical deviations from the probabilities expected for randomly generated integrals. We conclude therefore that the probabilistic first order attack is not as efficient as deterministic second order attacks and the probabilistic approach does not appear to be applicable to higher order integral attacks.

VII. CONCLUSIONS

We have considered the integral cryptanalysis of the BSPN cipher structure. For a 64-bit (or larger) BSPN, a first order integral attack is applicable to a 3 round cipher, requiring 2^{16} partial decryptions making use of 2^8 chosen plaintexts to determine a byte from the last round key. This is comparable to the general result discussed in [10], which tackles a general 3 round structure with an unknown S-box mapping. Extending integral cryptanalysis, it is possible to attack a 4 round BSPN structure using a second order attack (requiring 2^{16} chosen plaintexts and 2^{24} partial decryptions) or a probabilistic first order attack (requiring 2^{37} chosen plaintexts and 2^{53} partial decryptions). BSPN ciphers of 5 or more rounds appear to be secure against integral cryptanalysis.

ACKNOWLEDGMENT

The author thanks Huicong Liu, who did some preliminary investigative programming for this research.

REFERENCES

[1] A.M. Youssef, S.E. Tavares, and H.M. Heys, “A New Class of Substitution-Permutation Networks”, Proceedings of Workshop on Selected Areas in Cryptography (SAC ’96), Kingston, Ontario, Canada, pp. 132-147, 1996.

[2] E. Biham and A. Shamir, “Differential Cryptanalysis of DES-like Cryptosystems”, Proceedings of CRYPTO ’90, Lecture Notes in Computer Science, vol. 537, Springer, pp. 2-21, 1990.

[3] M. Matsui, “Linear Cryptanalysis Method for DES Cipher”, Proceedings of EUROCRYPT ’93, Lecture Notes in Computer Science, vol. 765, Springer, pp. 386-397, 1993.

[4] X. Zhang, H.M. Heys and C. Li, “Energy Efficiency of Encryption Schemes Applied to Wireless Sensor Networks”, Security and Communication Networks Journal, vol. 5, no. 7, Wiley, pp. 789-808, 2011.

[5] “Advanced Encryption Standard”, FIPS -197, National Institute of Standards and Technology, Nov. 2001.

[6] C.E. Shannon, “Communication Theory of Secrecy Systems”, Bell System Technical Journal, vol. 28, pp. 656-715, 1949.

[7] L.R. Knudsen and D. Wagner, “Integral Cryptanalysis”, Proceedings of Fast Software Encryption (FSE 2002), Lecture Notes in Computer Science, vol. 2365, Springer, pp. 112-127, 2002.

[8] J. Daemen, L. Knudsen, and V. Rijmen, “The Block Cipher SQUARE”, Proceedings of Fast Software Encryption (FSE 1997), Lecture Notes in Computer Science, vol. 1267, Springer, pp. 149-165, 1995.

[9] S. Lucks, “Attacking Seven Rounds of Rijndael Under 192-bit and 256-bit Keys”, Proceedings of 3rd AES Candidate Conference, pp. 215-229, 2000.

[10] A. Biryukov and A. Shamir, “Structural Cryptanalysis of SASAS”, Proceedings of EUROCRYPT 2001, Lecture Notes in Computer Science, vol. 2045, Springer, pp. 394-405, 2001.

[11] C. Cannière, A. Biryukov, and B. Preneel, “An Introduction to Block Cipher Cryptanalysis”, Proceedings of the IEEE, vol. 94, no. 2, IEEE, pp. 346-356, 2006.