K-Cipher: A Low Latency, Bit Length Parameterizable Cipher

Michael Kounavis, Sergej Deutsch, Santosh Ghosh, and David Durham

Intel Labs, Intel Corporation, 2111, NE 25th Avenue, Hillsboro, OR 97124
Email: {michael.e.kounavis, sergej.deutsch, santosh.ghosh, david.durham}@intel.com

Abstract— **We present the design of a novel low latency, bit length parameterizable cipher, called the "K-Cipher". K-Cipher is particularly useful to applications that need to support ultra low latency encryption at arbitrary ciphertext lengths. We can think of a range of networking, gaming and computing applications that may require encrypting data at unusual block lengths for many different reasons, such as to make space for other unencrypted state values. Furthermore, in modern applications, encryption is typically required to complete inside stringent time frames in order not to affect performance. K-Cipher has been designed to meet these requirements. In the paper we present the K-Cipher design and discuss its rationale. We also present results from our ongoing security analysis which suggest that only 2 to 4 rounds are sufficient to make the cipher operate securely. Finally, we present synthesis results from 2-round 32-bit and 64-bit K-Cipher encrypt datapaths, produced using Intel's ® 10 nm process technology. Our results show that the encrypt datapaths can complete in no more than 767 psec, or 3 clocks in 3.9-4.9 GHz frequencies, and are associated with a maximum area requirement of 1875 µm^2.**

I. Introduction

The paper presents a novel block cipher design which is lightweight, hardware efficient, as well as bit length parameterizable. Our cipher is called the "K-Cipher". In the K-Cipher design, the block length is not fixed (e.g., fixed to 128 bits), neither takes values from a small set of options (e.g., 64 or 128 bits). Instead the block length can take any arbitrary value between an upper and a lower bound and is an input parameter passed into the cipher.

The need for such block cipher comes from the requirements of modern applications. To address a wide range of vulnerabilities, applications employ cryptographic mechanisms to provide confidentiality and integrity. A common characteristic of applications is that their features are designed to work well with some existing or specially designed cipher (e.g., AES [2] or QARMA [3]).

In the paper we argue for the need of a new block cipher family that is more agile and easier to tune to the needs of a particular application or hardware. A new requirement we introduce is that the block length of a cipher should be an input parameter to the encryption operation. This should be the block length over which full confusion and diffusion operations are performed. Furthermore, the specification and security analysis of such block cipher should be, to some degree, block length independent. Such independence should substantially surpass what is accomplished today, where the block length is selected from a small number of options. Our design supports encryption at arbitrary block lengths which, in our software prototype prototype take all values from 24 to 1024 at increments of 1.

The reason why we believe such requirement is important, is because applications operate on a variety of data of different lengths. The encrypted portions of such data may be of arbitrary lengths as well. Rather than designing an application with a cipher of fixed block length in mind, we argue for the opposite. That is, to have the ability to arbitrarily tune the block length of a block cipher to meet the needs of a particular application or hardware.

Another property of our cipher family is that it supports ultra low latency encryption in hardware. Specifically, our 32-bit and 64-bit encrypt datapaths can complete in 3 clocks in 3.9-4.9 GHz frequencies, where datapaths are synthesized using Intel's ® 10 nm process technology. It is fair to state that the landscape of today's block ciphers, which are either standard (e.g., AES [1]), or are to be standardized though the current NIST lightweight cryptography competition [4], does not include any cipher that simultaneously meets the two requirements stated: (i) support for full confusion and diffusion over arbitrary block lengths, as part of the encryption and decryption operations; and (ii) support for ultra low latency encryption and decryption operations in hardware.

For example, past lightweight cipher designs such as NSA's "Simon" and "Speck" [5], or designs like "PRINCE" [6] are not bit length parameterizable. Furthermore, such ciphers support critical paths, which can get even smaller with the design proposed here. For example, the PRINCE rounds, even though contain simpler Sbox transformations, and simpler Mix Columns matrices when compared to AES, still require several clocks in the critical path, in typical client and server frequencies. Others ciphers like Simon employ a simple Feistel structure, which includes only elementary logical AND, XOR and rotation operations over 32-72 rounds. Furthermore, Simon supports only 5 fixed lengths: i.e., 32, 48, 64, 96 and 128 bits. Last, some submissions to the NIST lightweight cryptography competition, such as TinyJAMBU [8] or Xoodyak [9] support encryption at both very high speeds and arbitrary plaintext lengths. However, they do not fully diffuse the

978-1-7281-8086-1/20/$31.00

bits of their plaintext input into the bits of the ciphertext output. Instead, for a wide range of plaintext inputs, they merely add the input plaintext bits into the bits of some sponge state, in the finite field GF(2).

II. Overview

We envision that K-Cipher will be a useful tool for the encryption and decryption of data of varying lengths, performed by many different types of applications such as running in networking devices, gaming consoles, servers, low power clients and so on. K-Cipher supports fast encryption based on a novel confusion-diffusion network, which we discuss in this paper. The primitives it employs are: (i) block-wide addition with carries. In this operation, the carry-out bit is ignored. The operation is invertible, its inverse being subtraction with borrows. For the subtraction operation, K-Cipher just ignores the borrow-out bit; (ii) block-wide bit level reordering; and finally (iii) wide Sbox substitution, which is realized as inversion in a binary Galois Field.

K-Cipher is designed to support confidentiality at desired security levels by employing the least possible number of rounds r, which, in the proposed design is set by default to 2. There are also three additional modes of the cipher where $r = 3$, $r = 4$, and r is a configurable parameter. In the security analysis section of the paper, we discuss why we believe the range of 2-4 rounds is a good choice for the cipher.

Substitution box lengths are determined by the block length. For example, to diffuse across 32 bits, K-Cipher uses 8-bit substitution boxes, as the 32-bit length is smaller than the square of the 8-bit length, and the minimum number of rounds is 2. Similarly, to diffuse across 128 bits, K-Cipher uses 16-bit substitution boxes, as the 128-bit length is smaller than the square of the 16-bit length. All primitives of the K-Cipher apply to a wide range of block lengths. Arbitrary ciphertext lengths are supported using Galois field inverters of varying lengths, the sum of which is equal to the requested input and ciphertext lengths. In our implementation all employed Galois field inverters have fixed lengths but the last one, the length of which is determined by the block length.

III. K-Cipher Design

A. Notation

We will be denoting as $<a_{n-1} \dots a_1 a_0>$ a bit string of length n consisting of bits $a_0, \dots, a_{n-1}$, where a_0 is the least significant bit of the string and a_{n-1} is its most significant bit. Similarly, we will be denoting as $\mathrm{N}(<a_{n-1} \dots a_1 a_0>)$ the binary number which is represented by the string $<a_{n-1} \dots a_1 a_0>$.

B. The Aggressive Adder

Starting with Figure 1, we illustrate one of the basic components of the K-Cipher round called the "aggressive adder". The aggressive adder accepts as input some state, and then adds to this state a round key. The addition performed is not in the typical GF(2) arithmetic, but is in the integer arithmetic. Integer addition, if seen as a bit-logical operation, performs strong mixing of its input bits, in order to produce the bits of the output. The mixing performed demonstrates regularity due to the use of carry values. By "mixing" in this document we mean computations on single bit values that involve a plurality of AND, OR, NAND, NOR or XOR operations.

For example let's consider that we add the numbers $\mathrm{N}(<a_3 a_2 a_1 a_0>)$ and $\mathrm{N}(<b_3 b_2 b_1 b_0>)$ with each other and with some input carry value c_0. The first bit of the result is equal to $a_0 \oplus b_0 \oplus c_0$. The carry produced from the addition of the first two bits is equal to $a_0 b_0 \oplus b_0 c_0 \oplus a_0 c_0$. Similarly, the second bit of the result is $a_1 \oplus b_1 \oplus a_0 b_0 \oplus b_0 c_0 \oplus a_0 c_0$ and the carry produced from the addition of the second two bits is equal to $a_1 b_1 \oplus a_1 a_0 b_0 \oplus a_1 b_0 c_0 \oplus a_1 a_0 c_0 \oplus b_1 a_0 b_0 \oplus b_1 b_0 c_0 \oplus b_1 a_0 c_0$. Moving on to the addition of the third least significant bits of the input, the same pattern of computation is repeated. The input bits are XOR-ed with each other and with the input carry, in order to produce the output bit. Furthermore, the input bits are multiplied with each other in GF(2) arithmetic (i.e., undergo a logical AND operation) and with the input carry and, subsequently, the products are XOR-ed with each other in order to produce the output carry. The third least significant bit of the result, as computed using this pattern, is $a_2 \oplus b_2 \oplus a_1 b_1 \oplus a_1 a_0 b_0 \oplus a_1 b_0 c_0 \oplus a_1 a_0 c_0 \oplus b_1 a_0 b_0 \oplus b_1 b_0 c_0 \oplus b_1 a_0 c_0$. The third output carry is $a_2 b_2 \oplus a_2 a_1 b_1 \oplus a_2 a_1 a_0 b_0 \oplus a_2 a_1 b_0 c_0 \oplus a_2 a_1 a_0 c_0 \oplus a_2 b_1 a_0 b_0 \oplus a_2 b_1 b_0 c_0 \oplus a_2 b_1 a_0 c_0 \oplus b_2 a_1 b_1 \oplus b_2 a_1 a_0 b_0 \oplus b_2 a_1 b_0 c_0 \oplus b_2 a_1 a_0 c_0 \oplus b_2 b_1 a_0 b_0 \oplus b_2 b_1 b_0 c_0 \oplus b_2 b_1 a_0 c_0$.

From the logical expressions above, it becomes evident that the mixing performed by the addition with carries stage, as measured by the number of GF(2) products which are XOR-ed with each other, gets only stronger as one moves from the least significant bit of the result toward the most significant bit. In fact, it grows stronger exponentially. It is easy to show that the n-th output bit of the result is produced by XOR-ing $2^n + 1$ terms, of which $2^n - 1$ are products.

To destroy the regularity which characterizes the addition with the carries stage, the aggressive adder performs a bit level reordering operation on the addition output. Such reordering operation places the output bits coming from the integer adder in a seemingly random order, so that the number of GF(2) products of the logic equation of the result no longer increases monotonically but instead increases and decreases in a pseudorandom manner. Furthermore the bit level reordering operation aids the subsequent wide substitution stage, shown in Figures 2 and 3, ensuring that each bit of the output of the K-Cipher results from mixing all bits of the input with all bits of the key. The addition with carries is a bit length independent operation. Its specification is independent of the length of the inputs. It is also invertible, its inverse being the

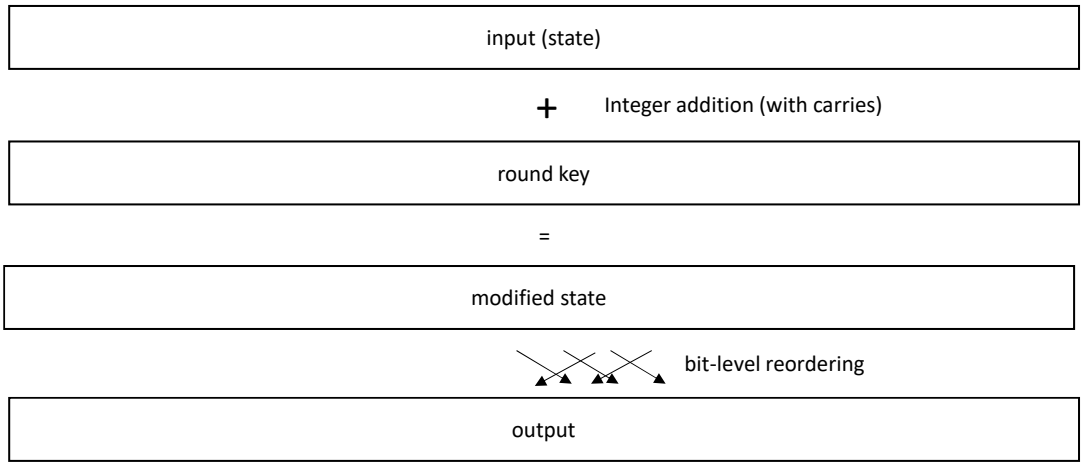

Fig. 1. The Aggressive Adder Component of the K-Cipher Round

subtraction with borrows. Any final carry-out or borrow-out signals produced from such operations are ignored.

C. Two Round *K*-Cipher Specification

Moving onto Figure 2, the processing steps of the default instance of the K-Cipher employing two rounds are shown. The first round consists of an aggressive adder stage that performs reordering using a first index sequence denoted by "reordering 1" and a first round key denoted by "Round Key 1". The aggressive adder of the first round is followed by a wide substitution stage referred to as "Sbox Layer" in the figure. The wide substitution stage is followed by yet another bit reordering stage, denoted by "reordering 2". The second round consists of an aggressive adder stage also performing reordering using a different round key and index sequence, denoted by "Round Key 2" and "reordering 3", respectively. The second aggressive adder is followed by a second wide substitution stage and bit reordering. The processing steps of the K-Cipher conclude with an XOR operation performed between the cipher state and a third round key denoted by "Round Key 3".

The Sbox layer performs the following steps. It first divides its input N bits into blocks of M bits. Let's assume for now that N is a multiple of M. The cases where N is not a multiple of M are discussed further below. If N is a multiple of M, the Sbox layer employs an array of N/M inverters in GF(2^M) arithmetic, which replace their input bits with the bits of the inverse in GF(2^M). The output undergoes yet another bit reordering operation, like the one employed by the aggressive adder, but this reordering uses a different index sequence. Inversion in the Galois Field arithmetic GF(2^M) is another operation supporting strong bit mixing. The mixing performed by the Galois Field inverters employed by the K-Cipher does not demonstrate the regularity of addition with carries and is in fact pseudo-random. K-Cipher is designed to support strong encryption security by employing additions and inversions in two unrelated types of arithmetic (i.e., Galois Field and integer) and by combining those into sequences of few rounds. Even though the additions and inversions may demonstrate imperfect differential distributions, their combination creates a much stronger cryptographic primitive. This aspect of our designed is further discussed in the analysis section below. Our experimental data suggest that K-Cipher rounds, despite the fact that they are few, succeed in strongly mixing their input and key bits potentially thwarting differential attacks.

The Sbox layer, as defined so far, is bit length independent provided that the length of the state of the cipher N is a multiple of the width of the inverters employed M. In this case, the specification of the cipher is generic and each wide substitution stage employs N/M inverters. If N is not a multiple M, then these situations can be handled as shown in Figure 3. In the figure there are m substitution boxes of width M which are employed, plus one more of width $K = N - m \cdot M$, where K is non-zero. The substitution stage employs m inverters in the in GF(2^M) arithmetic and one inverter in the GF(2^K) arithmetic handling the last K bits of the cipher state.

The generation of the index sequences employed by the K-Cipher in order to support bit level reordering is accomplished by the following algorithm: The algorithm first determines the number of times d it needs to iterate over the bits of a substitution box in order to distribute these bits over all substitution boxes, the number of which is b. We refer to these bits of a substitution box as "bits-to-be-reordered". The parameter d is equal to ceil(M/b). Then, for each of the d iterations, the algorithm generates a random sequence of numbers. These are the indexes of the substitution boxes where the bits-to-be-reordered associated with the current iteration will be placed. Sub-

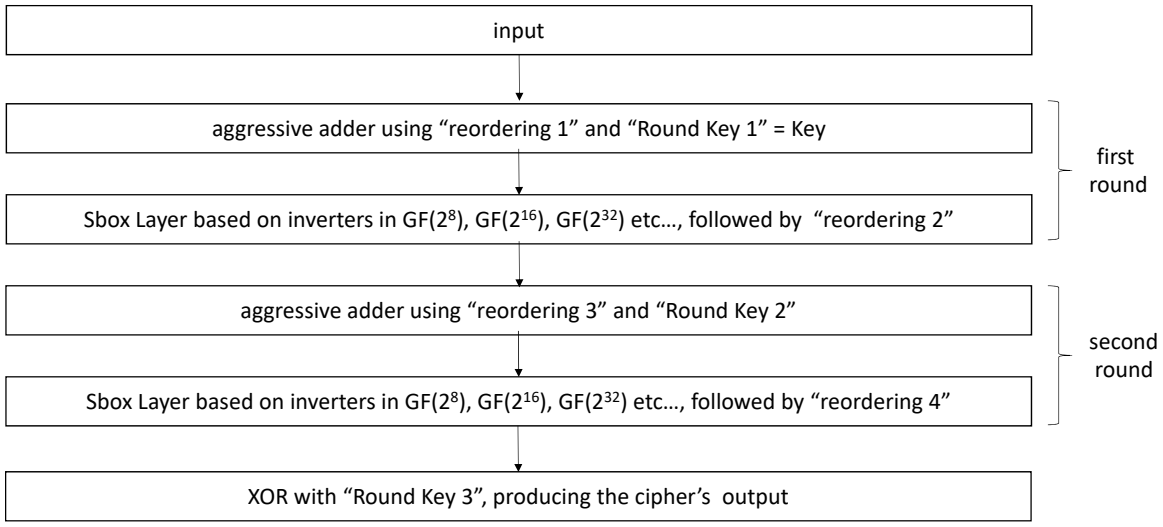

Fig. 2. Two Round K-Cipher Specification

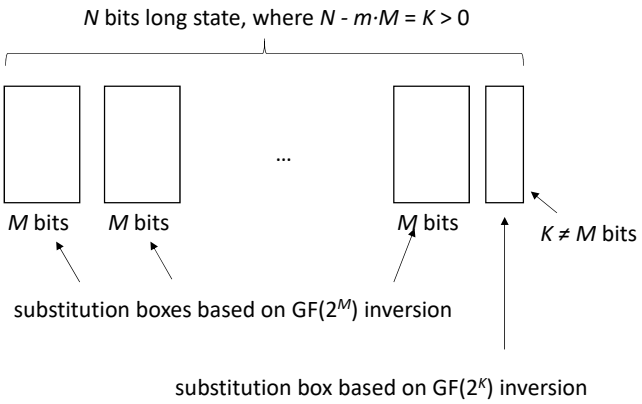

Fig. 3. Sbox Layer in which the length of the last box is different from the length of all other boxes

sequently, for each bit-to-be-reordered the algorithm picks a bit position at random from among the empty bit positions in the input bit's target substitution box and assigns this position to the bit. This last step is repeated for all iterations of a triply nested loop executed by the algorithm.

One can show that the algorithm produces sequences of indexes that are both "correct" and "proper". By correct we mean that every bit of the input is placed in a different bit position of the output and there is no input which is omitted from the output. By proper we mean that if such reordering operations are combined with wide substitution operations, then, after $\log_M N$ rounds all bits of the input have been fully mixed with each other, even if additions with carries are absent.

D. Key Schedule and Use of Tweaks

For block lengths less than 32, the cipher's key is equal to the key schedule. For larger block lengths a key schedule algorithm is employed. The key schedule algorithm of the K-Cipher has the same structure as the cipher itself shown in Figure 2. However, it uses a different set of reordering sequences. Furthermore, instead of adding key bits into its state, it adds a plurality of constants. This is done in order for the cipher to expand the key into a sequence of round keys used by the cipher rounds.

Last, we discuss how the K-Cipher can be turned into a tweakable block cipher. This is done as follows: As said above, the key schedule of the K-Cipher involves 3 or more round keys. A tweak value is accepted which has the same length as each round key. The tweak value is added to the first and the last round key of the K-Cipher key schedule using the aggressive adder algorithm of Figure 1. The result is a tweaked key schedule which is used by the encryption and decryption processes.

IV. Analysis of the K-Cipher

A. Security Analysis

To study the security of K-Cipher, we focus on the 24-bit encrypt version of this algorithm (Enc-24). This is because, for this version, the computation of the statistical properties of differentials applied on input ciphertexts is tractable and does not require any expensive computing infrastructure. As the specification of the cipher is bit length independent, our experimental findings for the 24-bit version of the cipher may also hint on the security properties of larger versions.

We first fix the encryption key to some random value and consider a fixed single bit input differential as well. For this

bit position of the differential	probability of the most dominant differentials			number of dominant, out of 16777216		
	$r = 2$	$r = 3$	$r = 4$	$r = 2$	$r = 3$	$r = 4$
0	$2^{-12.79}$	$2^{-18.42}$	$\mathbf{2^{-20.83}}$	541	422	**0**
2	$2^{-12.04}$	$2^{-17.68}$	$\mathbf{2^{-20.68}}$	456	490	**0**
4	$2^{-12.18}$	$2^{-18.09}$	$\mathbf{2^{-20.83}}$	318	392	**0**
6	$2^{-13.17}$	$2^{-18.91}$	$\mathbf{2^{-20.54}}$	380	111	**0**
8	$2^{-12.99}$	$2^{-18.61}$	$\mathbf{2^{-20.68}}$	362	299	**0**
10	$2^{-13.14}$	$2^{-18.96}$	$\mathbf{2^{-20.68}}$	500	231	**0**
12	$2^{-11.98}$	$2^{-17.61}$	$\mathbf{2^{-20.54}}$	619	639	**0**
14	$2^{-12.78}$	$2^{-18.51}$	$\mathbf{2^{-20.68}}$	366	361	**0**
16	$2^{-12.69}$	$2^{-18.07}$	$\mathbf{2^{-20.68}}$	469	413	**0**
18	$2^{-13.00}$	$2^{-18.12}$	$\mathbf{2^{-20.68}}$	472	352	**0**
20	$2^{-12.40}$	$2^{-17.48}$	$\mathbf{2^{-20.54}}$	677	590	**0**
22	$2^{-12.53}$	$2^{-18.00}$	$\mathbf{2^{-20.68}}$	656	639	**0**

Fig. 4. Statistical properties of the $2^{24} = 16,777,216$ differentials of the of the output of the K-Cipher Enc-24 algorithm, when the number of rounds r ranges from 2 to 4.

type of experiment we apply all possible plaintext values from 0 to $2^{24} - 1 = 16,777,215$. We repeat the process 16 times using different random keys, a total of $268,435,456$ encryption operations. We observe that differentials are almost random uniformly distributed with a statistical average of 1 and a standard deviation of 0.25. Furthermore, the probability of the most dominant differential is equal to $2^{-22.67}$, which is close to the ideal. Furthermore, the results are similar for other values of the fixed differential. This experiment indicates that any differential cryptanalysis on a K-Cipher Enc-24 oracle that uses a secret fixed key cannot extract any substantial information about the plaintext or key, unless the behavior of the cipher is studied for a plurality of different key values. Similar behavior is observed when we fix the key and plaintext and vary the input differential. In fact, in this case, output differentials are completely uniformly distributed.

To get a better understanding of the security of K-Cipher Enc-24, we consider another set of experiments where the plaintext and differential are both fixed and the key takes random values from the space of all possible key values. This space, for $r = 2$, is the set $\{0,1\}^{72}$, according to the cipher description presented above. Our results are shown in the table of Figure 4 and in the plot of Figure 5. In these experiments we consider that the key takes 16,777,216 different random values, and the experiments are repeated for all 24 possible single bit differentials, corresponding to a total of 402,653,184 encryption operations.

We observe that for $r = 2$, output differentials are still close to random uniformly distributed with a statistical average of 1 and standard deviation ranging between 2.67 and 8.13. However, this time, the histogram is populated with a small number of entries characterized by non-negligible frequencies. The probability associated with the highest of these frequencies ranges between $2^{13.17}$ and $2^{-11.98}$.

The total number of dominant entries is also shown in the table, where by "dominant" entries we mean entries demonstrating probability higher than 16 times the standard deviation, or a threshold approximately equal to $2^{N-6.6}$, with N being the block size of 24. We observe that dominant entries are indeed few, ranging between 318 and 677 out of 16,777,216. Because of the fact that these entries are few, and that they sparsely populate the histogram, as shown in Figure 5, any attack based on observing these dominant differentials should not do any better than any brute force attack on the key values.

If the number of rounds increases, K-Cipher Enc-24 demonstrates almost random uniformly distributed differentials and the dominant entries disappear. For $r = 3$, the probability of the most dominant differentials drops signifi-

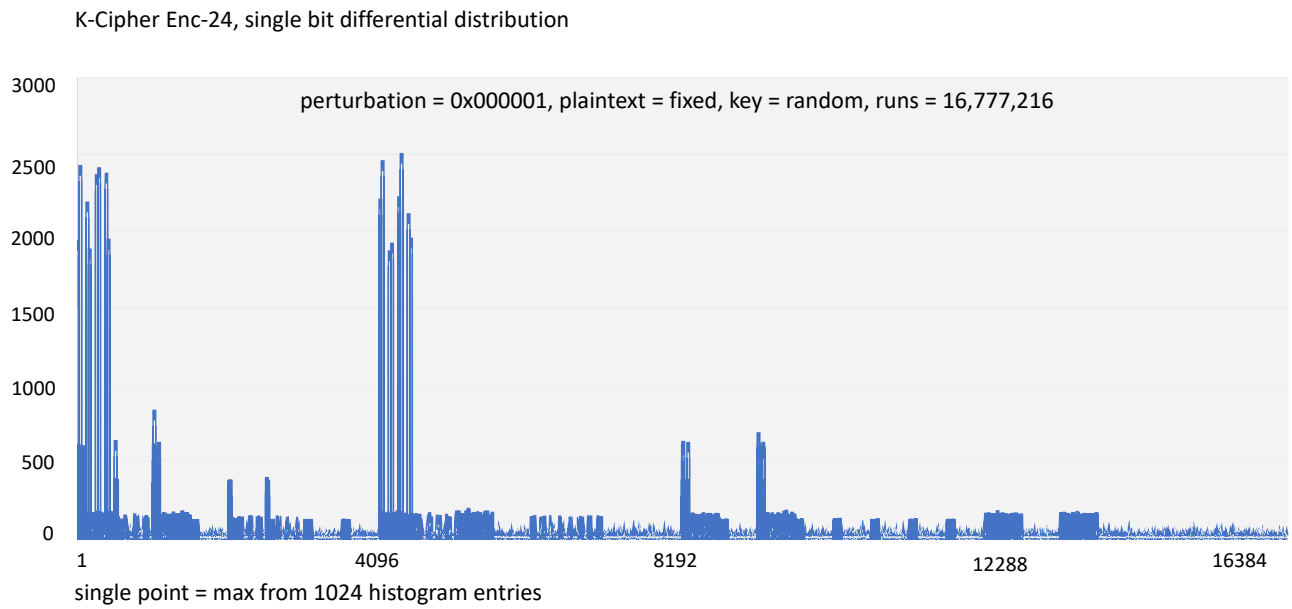

Fig. 5. Distribution of the $2^{24} = 16,777,216$ differentials of the of the output of the K-Cipher Enc-24 algorithm, when the input perturbation is equal to $0x000001$ and keys are randomly selected.

cipher	area (μm^2)	latency (psec)	number of clocks	freq.
K-Cipher Enc-32, $r = 2$	614	**613**	3	4.9 GHz
K-Cipher Enc-64, $r = 2$	1875	**767**	3	3.9 GHz

Fig. 6. Synthesis results and performance of the K-Cipher Enc-32 and K-Cipher Enc-64 algorithms, when the number of rounds r is equal to 2.

cantly to values between $2^{-18.96}$ and $2^{-17.48}$. Furthermore, if we add one more round, the histogram no longer demonstrates significant dominant entries. The probability of the most probable differentials does not exceed $2^{-20.54}$ which is close to ideal. It is these facts that make us believe that the range between 2 and 4 rounds is a good choice for the specification of the K-Cipher.

B. Performance

We have built a software prototype that supports encryption for all bit lengths from 24 to 1024 with increments of 1. Our optimized K-Cipher Enc-24 and K-Cipher Dec-24 routines used in the experiments above demonstrate a speed of 17 cycles per byte on an Intel ® Core ™ i7-8665U processor running at 1.9 GHz.

More interesting is the hardware performance of the K-Cipher. K-Cipher employs standard integer adders, substitutions that can be implemented using lookup tables, and reordering operations that can be implemented using wires. These components add minimal latency to the critical path. Moreover, the number of these components is small for $r \in 2, 3, 4$.

We have built and synthesized optimal encrypt and decrypt datapaths $r = 2$ and for the 32-bit and 64-bit versions of the K-Cipher family using Intel's ® 10 nm process technology. Our results are shown in the table of Figure 6. The area required by the encrypt and decrypt datapaths is 614 μm^2 and 1875 μm^2 respectively. Moreover, the complete encryption operation finishes in 617 psec and 767 psec respectively for the two datapaths, if there are no clocking constraints.

Finally, the implementations allow for the insertion of pipeline registers. The staged encryptions performed using these registers complete in 3 clocks. The minimum unconstrained clock periods supported correspond to frequency values of 4.9 and 3.9 GHz, respectively for the 32-bit and 64-bit datapaths.

V. Concluding Remarks

We presented the design of a new cipher, called the K-Cipher. K-Cipher is bit length parameterizable and only involves few low latency components in the critical path, specifically integer adders and Galois field inverters. As such, the K-Cipher can be a useful tool for developing secure applications without sacrificing performance. We have developed software and hardware prototypes of the cipher. The software prototypes can successfully perform encryptions and decryptions for all block lengths from 24 bits up to 1024 bits, at block length increments of one. The hardware prototypes demonstrate encryption in the order of few hundreds of psec.

As defined, the K-Cipher could also be used as part of other larger cryptographic constructions. Many different known cryptographic constructions could be employing the cipher, including Feistel structures, sponge structures, Davies Meyer constructions, modes such as CBC, CTR, XTS and so on. The cryptanalysis of the K-Cipher is ongoing. We also plan to post a more detailed specification document so that the community can further study this cipher.

References

[1] Advanced Encryption Standard (AES), Federal Information Processing Standards Publication FIPS PUB 197.

[2] Recommendation for Block Cipher Modes of Operation: The XTS-AES Mode for Confidentiality on Storage Devices, NIST Special Publication 800-38E.

[3] R. Avanzi, The QARMA Block Cipher Family, Cryptology ePrint Archive: Report 2016/444.

[4] NIST Lightweight Cryptography Competition, available online at https://csrc.nist.gov/projects/lightweight-cryptography

[5] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers The Simon and Speck Families of Lightweight Block Ciphers, Cryptology ePrint Archive: Report 2013/404.

[6] J. Borghoff, A. Canteaut, T. Guneysu, E. B. Kavun, M. Knezevic, L. R. Knudsen, G. Leander, V. Nikov, C. Paar, C. Rechberger, P. Rombouts, S. S. Thomsen, and T. Yalcin, PRINCE: a low-latency block cipher for pervasive computing applications, ASIACRYPT 2012, Proceedings of the 18th international conference on The Theory and Application of Cryptology and Information Security, Pages 208-225, Beijing, China — December 02 - 06, 2012.

[7] Y. Dodis, T. Liu, M. Stam, J. Steinberger, Indifferentiability of Confusion-Diffusion Networks, hskip 1em plus 0.5em minus 0.4emCryptology ePrint Archive: Report 2015/680.

[8] H. Wu, and T. Huang, TinyJAMBU: A Family of Lightweight Authenticated Encryption Algorithms, Submission to the NIST Lightweight Cryptography Competition, available online at https://csrc.nist.gov/CSRC/media/Projects/ Lightweight-Cryptography/documents/round-1/spec-doc/TinyJAMBU-spec.pdf.

[9] J. Daemen, S. Hoffert, M. Peeters, G. Van Assche, and R. Van Keer, Xoodyak, a lightweight cryptographic scheme, Submission to the NIST Lightweight Cryptography Competition, available online at https://csrc.nist.gov/CSRC/media/Projects/ Lightweight-Cryptography/documents/round-1/spec-doc/Xoodyak-spec.pdf.