The Hummingbird-2 Lightweight Authenticated Encryption Algorithm

Daniel Engels, Markku-Juhani O. Saarinen, and Eric M. Smith

REVERE SECURITY
4500 Westgrove Drive, Suite 335, Addison, TX 75001, USA.
{daniel.engels,mjos,eric.smith}@reveresecurity.com

Abstract. Hummingbird-2 is an encryption algorithm with a 128-bit secret key and a 64-bit initialization vector. Hummingbird-2 optionally produces an authentication tag for each message processed. Like it's predecessor Hummingbird-1, Hummingbird-2 has been targeted for low-end microcontrollers and for hardware implementation in light-weight devices such as RFID tags and wireless sensors. Compared to the previous version of the cipher, and in response to extensive analysis, the internal state has been increased to 128 bits and a flow of entropy from the state to the mixing function has been improved. In this paper we present the Hummingbird-2 algorithm, its design and security arguments, performance analysis on both software and hardware platforms, and timing analysis in relation to the ISO 18000-6C protocol.

Keywords: Hummingbird cipher, constrained devices, lightweight cryptography, ISO 18000-6C.

1 Introduction

Authenticated encryption algorithms provide confidentiality and integrity protection for messages using a single processing step. This results in performance and cost advantages, especially when the algorithm is implemented in hardware.

Hummingbird-2 is an authenticating encryption primitive that has been designed particularly for resource-constrained devices such as RFID tags, wireless sensors, smart meters and industrial controllers. Hummingbird-2 can be implemented with very small hardware or software footprint and is therefore suitable for providing security in low-cost ubiquitous devices.

The design described in this paper is an evolutionary step from Hummingbird-1 [8, 10, 11] and was developed in part as a response to the cryptanalysis of the cipher presented in [20]. Hummingbird-2 is resistant to all previously known cryptanalytic attacks.

The Hummingbird-2 does not directly fall to either traditional stream cipher or block cipher categories as it inherits properties from both. In this sense Hummingbird-2 resembles the Helix and Phelix proposals [9, 16, 22]. Since

Table 1. S-Boxes used in Hummingbird-2.

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$S_1(x)$	7	12	14	9	2	1	5	15	11	6	13	0	4	8	10	3
$S_2(x)$	4	10	1	6	8	15	7	12	3	0	14	13	5	9	11	2
$S_3(x)$	2	15	12	1	5	6	10	13	14	8	3	4	0	11	9	7
$S_4(x)$	15	4	5	8	9	7	2	1	10	3	0	14	6	12	13	11

Hummingbird-2 operates on 16-bit blocks, more efficiency can be realized in applications that chirp small messages, such as RFID devices or wireless sensors. This also makes it easy to layer in security in various protocol schemes.

This paper is structured as follows: A formal description of Hummingbird-2 is contained in Section 2. Section 3 has the preliminary results of cryptanalysis of the cipher. Software and hardware implementations are described in Section 4, together with ISO 18000-6C timing information and comparison in Section 5. Conclusions can be found in Section 6. Appendix A contains a set of implementation test vectors for Hummingbird-2.

2 Description of Hummingbird-2

The Hummingbird-2 cipher has a 128-bit secret key K and a 128-bit internal state R which is initialized using a 64-bit Initialization Vector IV. These variables are accessed as vectors of 16-bit words:

$$\begin{aligned} K &= (K_1, K_2, K_3, K_4, K_5, K_6, K_7, K_8), \\ R &= (R_1, R_2, R_3, R_4, R_5, R_6, R_7, R_8), \\ IV &= (IV_1, IV_2, IV_3, IV_4). \end{aligned}$$

Hummingbird-2 is entirely built from operations on 16-bit words: the exclusive-or operation on words ($\oplus$), addition modulo 65536 ($\boxplus$) and a nonlinear mixing function $f(x)$.

2.1 Nonlinear functions $f(x)$ and $\mathbf{WD16}(x, a, b, c, d)$

The nonlinear mixing function f consists of four-bit S-Box permutation lookups on each nibble of the word, followed by a linear mix.

The Hummingbird-2 S-Boxes S_1, S_2, S_3 and S_4 are given Table 1.[1] Let $S(x)$ denote the computation of four S-Boxes and $L(x)$ the linear transforma-

[1] Some early versions of Hummingbird-2 used a different set of S-Boxes from Serpent [1]. Hummingbird-2 was tweaked in May 2011 to use these S-Boxes.

tion which is expressed using the left circular shift (rotation) operator ($\lll$). We may write the f component as

$$\begin{aligned} S(x) =& S_1(x_0) \mid S_2(x_1) \mid S_3(x_2) \mid S_4(x_3) \\ L(x) =& x \oplus (x \lll 6) \oplus (x \lll 10) \\ f(x) =& L(S(x)). \end{aligned}$$

We further define a 16-bit keyed permutation WD16 using f as

$$\mathrm{WD16}(x, a, b, c, d) = f(f(f(f(x \oplus a) \oplus b) \oplus c) \oplus d). \tag{1}$$

The inverse of $f(x)$ and WD16 can be derived in straightforward fashion.

2.2 Initialization

The internal state of Hummingbird-2 is initialized with a four-round procedure using the 64-bit nonce IV. We first set

$$R^{(0)} = (IV_1, IV_2, IV_3, IV_4, IV_1, IV_2, IV_3, IV_4) \tag{2}$$

and then iterate for $i = 0, 1, 2, 3$ the following:

$$\begin{aligned} t_1 &= \mathrm{WD16}(R_1^{(i)} \boxplus \langle i \rangle, K_1, K_2, K_3, K_4) \\ t_2 &= \mathrm{WD16}(R_2^{(i)} \boxplus t_1, K_5, K_6, K_7, K_8) \\ t_3 &= \mathrm{WD16}(R_3^{(i)} \boxplus t_2, K_1, K_2, K_3, K_4) \\ t_4 &= \mathrm{WD16}(R_4^{(i)} \boxplus t_3, K_5, K_6, K_7, K_8) \\ R_1^{(i+1)} &= (R_1^{(i)} \boxplus t_4) \lll 3 \\ R_2^{(i+1)} &= (R_2^{(i)} \boxplus t_1) \ggg 1 \\ R_3^{(i+1)} &= (R_3^{(i)} \boxplus t_2) \lll 8 \\ R_4^{(i+1)} &= (R_4^{(i)} \boxplus t_3) \lll 1 \\ R_5^{(i+1)} &= R_5^{(i)} \oplus R_1^{(i+1)} \\ R_6^{(i+1)} &= R_6^{(i)} \oplus R_2^{(i+1)} \\ R_7^{(i+1)} &= R_7^{(i)} \oplus R_3^{(i+1)} \\ R_8^{(i+1)} &= R_8^{(i)} \oplus R_4^{(i+1)}. \end{aligned}$$

The initial state for encrypting the first plaintext word is $R^{(4)}$. Note that the two's complement numerical value of $i = 0 \ldots 3$ is used in computation of t_1.

2.3 Encryption

Encryption of a single word of plaintext P_i to ciphertext word C_i requires four invocations of WD16. [2]

$$
\begin{aligned}
t_1 &= \mathrm{WD16}(R_1^{(i)} \boxplus P_i, K_1, K_2, K_3, K_4) \\
t_2 &= \mathrm{WD16}(R_2^{(i)} \boxplus t_1, K_5 \oplus R_5^{(i)}, K_6 \oplus R_6^{(i)}, K_7 \oplus R_7^{(i)}, K_8 \oplus R_8^{(i)}) \\
t_3 &= \mathrm{WD16}(R_3^{(i)} \boxplus t_2, K_1 \oplus R_5^{(i)}, K_2 \oplus R_6^{(i)}, K_3 \oplus R_7^{(i)}, K_4 \oplus R_8^{(i)}) \\
C_i &= \mathrm{WD16}(R_4^{(i)} \boxplus t_3, K_5, K_6, K_7, K_8) \boxplus R_1^{(i)}.
\end{aligned}
$$

After each encrypted / decrypted word, we perform the following state update:

$$
\begin{aligned}
R_1^{(i+1)} &= R_1^{(i)} \boxplus t_3 \\
R_2^{(i+1)} &= R_2^{(i)} \boxplus t_1 \\
R_3^{(i+1)} &= R_3^{(i)} \boxplus t_2 \\
R_4^{(i+1)} &= R_4^{(i)} \boxplus R_1^{(i)} \boxplus t_3 \boxplus t_1 \\
R_5^{(i+1)} &= R_5^{(i)} \oplus (R_1^{(i)} \boxplus t_3) \\
R_6^{(i+1)} &= R_6^{(i)} \oplus (R_2^{(i)} \boxplus t_1) \\
R_7^{(i+1)} &= R_7^{(i)} \oplus (R_3^{(i)} \boxplus t_2) \\
R_8^{(i+1)} &= R_8^{(i)} \oplus (R_4^{(i)} \boxplus R_1^{(i)} \boxplus t_3 \boxplus t_1).
\end{aligned}
$$

A shorthand of this is $C = E(P)$. The state variable R is stepped by one iteration for each invocation of E. Note that the update function can be simplified since certain terms are re-used.

2.4 Authenticating fixed-length unencrypted associated data

Authenticated Encryption with Associated Data (AEAD) is a method of using Hummingbird that encrypts / decrypts a payload and also authenticates any associated data (AD) that travels alongside the ciphertext such as the nonce and a packet header. AD processing is optional in implementations.

AD processing occurs after the entire encrypted payload has processed. We simply compute $E(A_i)$ but transmit A_i itself instead. Note that the size of the AD must be fixed (known by the recipient).

[2] Some early versions of this paper had a typographic error here as the final addition of $R_1^{(i)}$ was missing. Thanks to Jean-Philippe Aumasson for spotting this.

2.5 Stream cipher mode: A technique for encoding short fixed-length fields

Sometimes it is desirable to communicate, without message expansion, datagrams which are less than 16 bits in size. We describe an encoding technique which enables this.

Let x be the short message of $1 \leq n \leq 15$ bits. The ciphertext message is derived from the n least significant bits of $x \oplus E(0)$. If message integrity is required, the state is further updated by $E(x)$. Decoding is straightforward.

Note that encrypting the two words 0 and x has exactly the same effect on state and hence special care must be taken to ensure that both parties are in sync. If a protocol requires arbitrary-length authenticated messages, this technique can be used for padding, but the total message length (in bits) must be specified and verified in an unambiguous fixed-length AD field.

2.6 Computing the Message Authentication Code

To compute a message authentication tag T of $n \leq 8$ words, we first finalize the message by first stepping the cipher three times without producing any output.

$$E(IV_1 \boxplus R_1 \boxplus R_3 \boxplus n)$$
$$E(IV_2 \boxplus R_1 \boxplus R_3)$$
$$E(IV_3 \boxplus R_1 \boxplus R_3).$$

Here R_1 and R_3 denotes the contents of those register words immediately before each invocation of E. We then construct the words of the authentication tag T as follows:

$$\begin{aligned} T_1 &= E(IV_4 \boxplus R_1 \boxplus R_3) \\ T_i &= E(R_1 \boxplus R_3) \quad \text{for i} = 2, 3, \ldots, \text{n}. \end{aligned}$$

2.7 Uniqueness Requirement for IVs and Keys

Hummingbird-2 is an authenticated encryption primitive and may be used in similar fashion as the Galois/Counter Mode (GCM) and GMAC, which are part of NSA "Suite B" algorithms. Implementers would be wise to take similar care in ensuring that keys and IVs are never repeated. We restate the requirement from Section 8 of [7].

> The probability that the authenticated encryption function ever will be invoked with the same IV and the same key on two (or more) distinct sets of input data shall be no greater than 2^{-32}.

Generally speaking, compliance to this requirement is recommended in order to mitigate the risk of using Hummingbird-2 in certain applications, and this cipher is not as vulnerable to repeated IV attacks as AES-GCM and should be able to resist many realistic chosen-IV attacks.

Indeed Hummingbird-2 leaks very little information should the nonce be repeated. At most all you get are exact repetitions in the ciphertext where you have exact repetitions in the plaintext.

3 Development and Analysis

The main differences between Hummingbird-1 and Hummingbird-2 are:

- The key size has been set to 128 bits to be commensurable with the actual security of the cipher.
- The state size of the cipher has been increased from 80 bits to 128 bits and the LFSR has been eliminated. The last four new state registers R_5, R_6, R_7, and R_8 are called the "accumulator" registers.
- The keyed transform, called the "E Box" in [20] now only has four invocations of the S-Boxes, compared to five in Hummingbird-1. This increases the encryption speed of the cipher.
- The authentication mechanism has been improved to thwart a message extension attack.
- Support for authenticating unencrypted associated data has been introduced with the AEAD mode discussed in Section 2.4.
- Recommendations to the reuse of keys and IVs have been introduced.
- An important design criteria was compatibility with the ISO 18000-6C timing requirements as discussed in Section 5.

We note that prior to publication, Hummingbird-2 has been subjected to a significant cryptanalytic security assurance effort. For this, the services of Jim Frazer & Son Cryptography (formerly ISSI) and the U. Waterloo Centre for Applied Cryptography Research were used, in addition to input from public analysis of Hummingbird-1. After thousands of hours of cryptanalysis, no significant flaws or sub-exhaustive attacks against Hummingbird-2 have been found. We summarize the results of our analysis in the following sections.

3.1 Structure of the Cipher

Hummingbird-2 has been designed to be as light-weight as possible while still maintaining a reasonable security margin against attacks. The state size of 128 bits should not be confused with the state size of stream ciphers (where the

key is usually loaded into the evolving state registers upon initialization). In Hummingbird-2 the key is kept constant and hence we could say that the state is 128+128=256 bits of which 128 bits is evolving. There is strong experimental and theoretical evidence that the cycle upon constant input words is 2^{127}.

The initialization function is one-to-one from the IV to the four state registers R_1, R_2, R_3, and R_4. Hence there are no nonce collisions. If more than 2^{64} invocations of the E function is performed under a single key, a birthday condition in the internal state may occur. The usage condition given in Section 2.7 will prevent this. Such a birthday condition has very limited cryptographic implication beyond serving as a distinguisher with that complexity. We note that this is the same bound that can be found for the AES algorithm [17].

3.2 S-Box selection

The S-Box (Table 1 and Section 2.1) selection was based on specific research presented in more detail in [21]. The S-Boxes also belong to the optimal classes discussed in [14].

We performed an exhaustive search through $16!$ possible permutations. The S-Boxes specifically belong to a classes that ideally satisfy the following conditions:

- Optimal differential bound $p \leq 1/4$, linear bound $\epsilon \leq 1/4$, and branch number 3.
- There is a minimum number of differential characteristics and linear approximations at the bounds.
- All S-boxes belong to a different linear equivalence class.
- The four S-boxes have a large Hamming distance from each other and the identity permutation.
- The algebraic degree of all but one output bit is 3 and each output bit is nonlinearly dependent on a maximum number of input bits.

3.3 Differential cryptanalysis

Hummingbird-2 has been designed to be resistant to Differential Cryptanalysis [3, 4]. The most interesting findings in our research involve the high-bit differential $\Delta = \texttt{8000}$ which behaves in the same way under modular addition and XOR. Much of the nonlinearity for lower bits comes from interplay of these two operations. The four rotations in the initialization phase were introduced to increase the resistance of the cipher against certain related-key attacks.

We have verified that Hummingbird-2 is provably resistant against the types of attacks described by Saarinen in [20]. This was done by performing a search

of all high-bit differentials in both initialization and actual encryption phases of the cipher.

Differentials in the encryption function Let $\mathtt{H} = \mathtt{8000}$ denote the high bit differential and x some undefined ciphertext differential. We found the following differentials that hold with probability 1.

$$\begin{aligned}
&\Delta P = \mathtt{0}\ \ \Delta R = (\mathtt{0\,0\,0\,H\,0\,0\,0\,0}) \ \Rightarrow\ \Delta C = \mathtt{x}\ \ \Delta R = (\mathtt{0\,0\,0\,H\,0\,0\,0\,H})\\
&\Delta P = \mathtt{H}\ \ \Delta R = (\mathtt{H\,0\,0\,0\,0\,0\,0\,0}) \ \Rightarrow\ \Delta C = \mathtt{H}\ \ \Delta R = (\mathtt{H\,0\,0\,H\,H\,0\,0\,H})\\
&\Delta P = \mathtt{H}\ \ \Delta R = (\mathtt{H\,0\,0\,H\,0\,0\,0\,0}) \ \Rightarrow\ \Delta C = \mathtt{x}\ \ \Delta R = (\mathtt{H\,0\,0\,0\,H\,0\,0\,0})\\
&\Delta P = \mathtt{0}\ \ \Delta R = (\mathtt{0\,H\,H\,0\,H\,0\,0\,0}) \ \Rightarrow\ \Delta C = \mathtt{0}\ \ \Delta R = (\mathtt{0\,H\,H\,0\,H\,H\,H\,0})\\
&\Delta P = \mathtt{H}\ \ \Delta R = (\mathtt{H\,H\,H\,0\,H\,0\,0\,0}) \ \Rightarrow\ \Delta C = \mathtt{H}\ \ \Delta R = (\mathtt{H\,H\,H\,H\,0\,H\,H\,H})\\
&\Delta P = \mathtt{H}\ \ \Delta R = (\mathtt{H\,H\,H\,H\,H\,0\,0\,0}) \ \Rightarrow\ \Delta C = \mathtt{x}\ \ \Delta R = (\mathtt{H\,H\,H\,0\,0\,H\,H\,0})\\
&\Delta P = \mathtt{0}\ \ \Delta R = (\mathtt{0\,H\,H\,H\,H\,0\,0\,0}) \ \Rightarrow\ \Delta C = \mathtt{x}\ \ \Delta R = (\mathtt{0\,H\,H\,H\,H\,H\,H\,H}).
\end{aligned}$$

It can be observed that these differentials can't be used to construct an iterative differential.

Related keys in encryption For two related keys we found one iterative differential which holds with probability one. When the two keys are related by $\Delta K = (\mathtt{H\,0\,0\,0\,H\,0\,0\,0})$ we have:

$$\Delta P = \mathtt{H}\ \ \Delta R = (\mathtt{0\,0\,0\,0\,H\,0\,0\,0}) \ \Rightarrow\ \Delta C = \mathtt{x}\ \ \Delta R = (\mathtt{0\,0\,0\,0\,H\,0\,0\,0}).$$

The ciphertext differential $\Delta C = x$ has some nontrivial value. We haven't found a direct way to exploit this related key property in an attack.

3.4 Linear and Algebraic Cryptanalysis

Hummingbird-2 has been designed to be resistant to Linear Cryptanalysis [15]. We performed a search for best linear masks in the mixing function f. Encryption of a single plaintext word involves sixteen invocations of f, five additions of state words (R_1, R_2, R_3, R_4, and R_1 again) and XOR keying with both static keys and dynamic accumulator variables.

We ignore the modular additions in our analysis. The search and probability calculation was performed on up to four invocations of f, which is no longer distinguishable. Our findings give significant confidence to assert that Hummingbird-2 is resistant to linear cryptanalysis up to twelve rounds of f. We are working to use multiple linear approximations in our analysis [2].

The algebraic degree and branch number of the S-Boxes alone thwarts most forms of algebraic distinguishing attacks such as Cube Testers [6] and d-monomial distinguisers [19]. A typical black-box chosen-IV scenario is made difficult by the rather complicated initialization routine that has a total of sixteen WD16 invocations.

4 Implementation and Performance

Hummingbird-2 has been implemented in hardware and in software for various microcontroller architectures. Functions were written in assembly language and hand-optimized for all platforms. All library functions are C-callable and many development environments are supported.

4.1 Microcontroller Software Implementations

As Hummingbird-2 algorithm allows for trade-offs between implementation speed and size, we have implemented up to three software implementation profiles for some microcontroller platforms. Table 2 gives the characteristics of these implementations.

Table 2. Microcontroller software implementations of Hummingbird-2. Encryption and decryption speeds are given in cycles per 16-bit word.

Target	**Encr.**	**Decr.**	**Size**	**MAC-64**	**Init.**	**RAM** $\approx$
MSP430 "Tiny"	1520	1544	770	10768	5984	50
MSP430 "Fast"	576	729	2518	4101	2187	114
MSP430 "Furious"	359	560	3648	2648	1361	114
AVR "Fast"	745	930	3600	5689	2970	114
AVR "Furious"	574	770	4178	4310	2139	114
AVR "Ultimate"	495	652	3200	3764	1800	1500
PIC24 "Fast"	319	371	2227	2248	1162	114
PIC24 "Furious"	271	362	4959	1897	912	114
ARM Cortex	332	336	2200	2525	1492	116

4.2 Hardware Implementations

Three hardware implementation profiles have been produced with differing size, power and speed characteristics. Table 3 summarizes these implementations. Figure 4.2 shows the layouts of the three cores.

– High Performance Design HB2-ee4c. This 4 clock implementation of Hummingbird is targeted at fast encryption (4 clocks) and maximum throughput and bandwidth.
– Low Area and Power Design HB2-ee16c. This 16 clock implementation of Hummingbird is targeted at minimum area and power.
– Ultra Low Area and Power Design HB2-ee20c. This 20 clock implementation of Hummingbird is targeted at ultra low area and power.

The process used was TSMC 0.13 μm, operating with 1.2 V. Encryption speed can be derived by dividing the operating frequency by the number of clocks required to encrypt a single word.

Table 3. Hardware implementations of Hummingbird-2. Encryption and decryption speeds are given in cycles per 16-bit word. The library used was the TSMC 180nm, 6 level metal, high density. Synopsis synthesis tools were used.

Profile	**Frequency**	**Clocks per word**	**Peak pwr (μW)**	**Leakage (μW)**	**Area (μm^2)**	**Gate Equiv.**
HB2-ee4c	100 kHz	4	1.93	4.17	27381	3220
HB2-ee4c	10 MHz	4	163.1	4.17	27381	3220
HB2-ee16c	100 kHz	16	1.845	2.85	20871	2332
HB2-ee16c	10 MHz	16	156.8	2.85	20871	2332
HB2-ee20c	100 kHz	20	1.73	2.63	19383	2159
HB2-ee20c	10 MHz	20	149.1	2.63	19383	2159

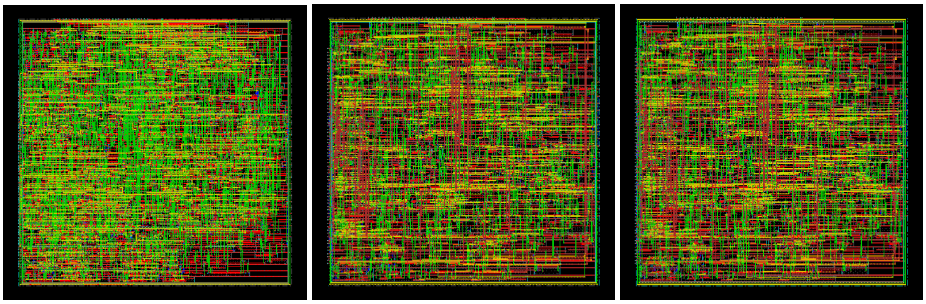

Fig. 1. From left: The layouts of HB2-ee4c, HB2-ee16c, and HB2-ee20c in 0.13 μm process.

5 HB2 Timing Compatibility with ISO 18000-6C

The ISO 18000-6C protocol [12] is the leading passive UHF RFID protocol in terms of number of tags sold today. The 18000-6C protocol specifies only a 32-bit password for access control and an electronic kill function.

The primary functionality of the 18000-6C protocol is for fast and efficient tag identification across a range of operating environments. Consequently, 18000-6C supports a range of data rates at which the interrogator and the tag may communicate. Communications are controlled by the interrogator in a Reader Talks First communication scheme. The interrogator begins a communication sequence by issuing a preamble that defines the length of the logic 0 and logic 1 symbols. The length of the logic 0 symbol is referred to as Tari, which is a fundamental timing parameter for communications.

For all commands except the Write command, the Tari value determines the amount of time the tag has to begin its response to the reader after the reader has completed the last symbol in its command to the tag. This time is referred to as T1 time. Table 4 shows the T1 timing for the minimum Tari value of 6.25 μs, the maximum Tari value of 25 μs, and a commonly used Tari value of 12.5 μs.

Table 4. T1 Timing Values and Available Clock Cycles.

Tari (μs)	T1 Time (μs)	Cycles 1.5 MHz	Cycles 2 MHz	Cycles 2.5 MHz
6.25	39.06	58	78	97
12.5	78.125	117	156	195
25	187.5	281	375	468

In addition to the T1 timings, Table 4 shows the number of full clock cycles available for computation within T1 for three on tag clock frequencies around 2 MHz, a common on-tag clock frequency.

Table 5 compares the clocks per bit required to encrypt a single block for various ciphers that can be implemented with up to approximately three thousand gate equivalents and are therefore fit for RFID use. The figures for Katan have been derived from [5], for Present, ICEBERG and AES from [18] and for Trivium and Grain from [13].

Based upon the clocks per bit for each cipher, in Table 6 we compare the number of clocks required to encrypt various amounts of data.

Command decode and processing overhead may be considerable. Furthermore, the mode overhead and initialization of the basic block ciphers is not

Table 5. Clocks Per Bit.

Cipher	Block Size (bits)	Key Size (bits)	Clocks Per Bit
HB2	16	128	0.25
Grain-128	1	128	1
Trivium	1	128	1
Present-80	64	80	0.5
Present-128	64	128	0.5
Katan32	32	80	8
Katan48	48	80	5.31
Katan64	64	80	3.98
Iceberg	64	128	0.25
AES-128	128	128	1.25

Table 6. Comparison of Clock Cycles to Encrypt. Note that most block ciphers require initialization every time key is changed.

Cipher	Init	16 bits	32 bits	48 bits	64 bits	96 bits	128 bits
HB2	16	4	8	12	16	24	32
Grain-128	513	16	32	48	68	96	128
Trivium	1333	16	32	48	68	96	128
Present-80	0*	32	32	32	32	64	64
Present-128	0*	32	32	32	32	64	64
Katan32	0*	256	256	512	512	768	1024
Katan48	0*	255	255	255	510	510	765
Katan64	0*	255	255	255	255	510	510
Iceberg	0*	16	16	16	16	32	32
AES-128	0*	160	160	160	160	160	160

considered. This additional processing will require even more overhead. While some operations may be performed in parallel, the command decode and processing must be performed prior to any cryptographic functions being performed.

In conclusion, HB2 is well suited for use in passive RFID systems due to its low power consumption, which minimally impacts range, and its high speed that enables the tag to continue normal operation within T1 timings.

6 Conclusions

We have presented Hummingbird-2, a light-weight authenticated encryption algorithm that we believe to be resistant to all standard attacks to block ciphers and stream ciphers such as differential and linear cryptanalysis, structure attacks and various algebraic attacks. Hummingbird-2 also has the further advantage of being resistant to chosen-IV attacks.

We have also presented results of software and hardware implementations of Hummingbird-2. Hummingbird-2 can be implemented with little more than 2000 gate equivalents, making it well suited for ubiquitous devices such as RFID tags and sensors. Hummingbird-2 has the additional advantage over other lightweight encryption primitives that it produces a message authentication code.

Acknowledgements. In addition to the anonymous RFIDSec '11 program committee members, we would like to thank Whitfield Diffie (who designed the original WD16 function), Peter Schweitzer and the members of CACR and ISSI teams. Troy Hicks and Ken Lauffenburger were behind the hardware work on Hummingbird-2. Jared Smothermon, Stanford Hudson, and Bob Nimon did the Software implementations on various platforms.

References

1. R. ANDERSON, E. BIHAM AND L. KNUDSEN. "Serpent: A Proposal for the Advanced Encryption Standard." http://www.cl.cam.ac.uk/~rja14/Papers/serpent.pdf, 1999.
2. A. BIRYUKOV, C. DE CANNIÈRE AND M. QUISQUATER. "On Multiple Linear Approximations." CRYPTO 2004, LNCS 3152, Springer, pp. 1-22, 2004.
3. E. BIHAM AND A. SHAMIR. "Differential Cryptanalysis of DES-like cryptosystems." In A. Menezes and S.A. Vanstone (Eds.): CRYPTO 1990. LNCS 537, pp. 2–21. Springer (1990)
4. E. BIHAM AND A. SHAMIR. "Differential Cryptanalysis of the Data Encryption Standard." Springer (1993)
5. C. DE CANNIÈRE, O. DUNKELMAN AND M. KNEŽEVIĆ. "KATAN & KTANTAN – A Family of Small and Efficient Hardware-Oriented Block Ciphers." CHES 2009, LNCS 5747, Springer, pp. 272–288, 2009.
6. I. DINUR AND A. SHAMIR. "Cube Attacks on Tweakable Black Box Polynomials." EUROCRYPT 2009, LNCS 5479, Springer, pp. 278–299, 2009.
7. M. DWORKIN. "Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC." NIST Special Publication 800-38D, November, 2007.
8. X. FAN, H. HU, G. GONG, E. M. SMITH AND D. ENGELS. "Lightweight Implementation of Hummingbird Cryptographic Algorithm on 4-Bit Microcontroller." The 1st International Workshop on RFID Security and Cryptography 2009 (RISC'09), pp. 838–844, 2009.
9. N. FERGUSON, D. WHITING, B. SCHNEIER, J. KELSEY, S. LUCKS, AND T. KOHNO "Helix: Fast Encryption and Authentication in a Single Cryptographic Primitive." FSE 2003, LNCS 2887, Springer, pp. 330–346, 2003.
10. D. ENGELS, X. FAN, G. GONG, H. HU AND E. M. SMITH. "Ultra-Lightweight Cryptography for Low-Cost RFID Tags: Hummingbird Algorithm and Protocol." Centre for Applied Cryptographic Research (CACR) Technical Reports, CACR-2009-29. http://www.cacr.math.uwaterloo.ca/techreports/2009/cacr2009-29.pdf
11. D. ENGELS, X. FAN, G. GONG, H. HU AND E. M. SMITH. "Hummingbird: Ultra-Lightweight Cryptography for Resource-Constrained Devices." 1st International Workshop on Lightweight Cryptography for Resource-Constrained Devices (WLC'2010). Tenerife, Canary Islands, Spain, January 2010

12. INTERNATIONAL STANDARDIZATION ORGANIZATION. "ISO/IEC 18000-6:2010. Information technology – Radio frequency identification for item management – Part 6: Parameters for air interface communications at 860 MHz to 960 MHz."
13. T. GOOD AND M. BENAISSA. "Hardware results for selected stream cipher candidates." eStream, ECRYPT Stream Cipher Project Report 2007 / 023. Proceedings of SASC 2007. 2007.
14. G. LEANDER AND A. POSCHMANN. "On the Classification of 4 Bit S-Boxes." In C. Carlet and B. Sunar (Eds.): WAIFI 2007, LNCS 4547, pp. 159–176. Springer (2007)
15. M. MATSUI. "Linear cryptoanalysis method for DES cipher." In T. Helleseth (Ed.): EUROCRYPT 1993. LNCS 765, pp. 386–397. Springer (1993)
16. F. MULLER. "Differential Attacks against the Helix Stream Cipher." FSE 2004, LNCS 3017, Springer, pp. 94–108, 2004.
17. NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. "The Advanced Encryption Standard (AES)." FIPS Publication 197, U.S. DoC/NIST, November 26, 2001.
18. A. POSCHMANN "Lightweight Cryptography - Cryptographic Engineering for a Pervasive World". PhD Thesis. Europaeischer Universitaetsverlag, in the IT-Security series, no 8. ISBN 978-3899663419. Also available as IACR ePrint 2009/516, 2009.
19. M.-J. O. SAARINEN. "d-Monomial Tests are Effective Against Stream Ciphers." State of the Art in Stream Ciphers (SASC) 2006. Workshop Record, K.U. Leuven, 2006.
20. M.-J. O. SAARINEN. "Cryptanalysis of Hummingbird-1." FSE 2011, LNCS 6733, Springer, pp. 328–341, 2011.
21. M.-J. O. SAARINEN. "Cryptographic Analysis of All 4×4 - Bit S-Boxes." Submitted for publication. Available as IACR ePrint 2011/218, 2011.
22. D. WHITING, B. SCHNEIER, S. LUCKS, AND F. MULLER. "Phelix – Fast Encryption and Authentication in a Single Cryptographic Primitive." ECRYPT Stream Cipher Project Report 2005/027. `http://www.schneier.com/paper-phelix.html`, 2005.

A Test Vectors

The test data is given as an array of bytes. When using these test vectors, note that Hummingbird-2 processes data in little-endian fashion (this means that the first 16-bit plaintext word in the second test vector is actually `0x1100`).

```
Secret key   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
IV / Nonce   00 00 00 00 00 00 00 00
Plaintext    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Ciphertext   C4 EF 87 A8 4F 05 A9 91 57 46 44 81 6E 25 3A CF
MAC          BA ED 40 F0 67 B0 E1 3C 76 F3 59 41 A2 B2 D1 35

Secret key   01 23 45 67 89 AB CD EF FE DC BA 98 76 54 32 10
IV / Nonce   12 34 56 78 9A BC DE F0
Plaintext    00 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF
Ciphertext   5B D1 F8 AD 23 14 20 F4 BA B1 54 C2 45 29 3D 38
MAC          C4 F6 74 C0 F6 4B 21 E7 37 24 DC 76 A6 6C 39 19
```

Document version 20110527203200.