JITEL (Jurnal Ilmiah Telekomunikasi, Elektronika, dan Listrik Tenaga) p-ISSN: 2774-7972
Vol. 1, No. 2, September 2021, pp. 175-188 e-ISSN: 2775-6696
DOI: https://doi.org/10.35313/jitel.v1.i2.2021.175-188

Implementasi algoritma *block cipher four* pada mikrokontroler STM32F103C8T6

Muhammad Adli Rizqulloh[1*], Yoyo Somantri[2], Resa Pramudita[3], Agus Ramelan[4]
[1,3]Program Studi Pendidikan Teknik Otomasi Industri dan Robotika, Universitas Pendidikan Indonesia
[2]Program Studi Pendidikan Teknik Elektro, Universitas Pendidikan Indonesia
Jl. Dr. Setiabudi No. 229, Kota Bandung, Jawa Barat 40154, Indonesia
[4]Program Studi Teknik Elektro, Universitas Sebelas Maret
Jl. Ir. Sutami No. 36, Kota Surakarta, Jawa Tengah 57126, Indonesia
[1*]muhammad.adli.riz@upi.edu, [2]ysomantri@upi.edu, [3]resa.pd@upi.edu, [4]agusramelan@staff.uns.ac.id

ABSTRAK

Pada masa industri 4.0, data menjadi salah satu komponen yang wajib dilindungi. *Block cipher* merupakan salah satu algoritma yang digunakan untuk mengamankan data. Penelitian ini bertujuan untuk mengimplementasikan algoritma *block cipher four* (BCF) pada mikrokontroler. Parameter yang menjadi tolak ukur antara lain besaran *flash* dan RAM mikrokontroler yang terpakai, serta kecepatan eksekusi proses komputasi algoritma BCF. Mikrokontroler akan menjalankan algoritma BCF dengan urutan komputasi *key-schedule*, enkripsi, dan dekripsi. Setiap kali memulai proses komputasi, maka pin *trigger* pada mikrokontroler akan mengirimkan sinyal *rising* ke osiloskop dan pada saat selesai melakukan komputasi maka pin *trigger* mikrokontroler akan mengirimkan sinyal *falling* ke osiloskop. Hasil penelitian menunjukkan algoritma BCF dapat diimplementasikan pada mikrokontroler STM32F103C8T6. *Flash* dan RAM yang digunakan mencapai 22,02 Kb dan 5,12 Kb. Algoritma BCF yang diimplementasikan pada mikrokontroler STM32F103C8T6 mampu berjalan sampai dengan 704 kali lebih cepat jika dibandingkan dengan prosesor NIOS II, 11 kali lebih cepat dibandingkan dengan AES-*Engine*, dan lebih lambat 4 kali jika dibandingkan dengan BCF-*Engine*.

Kata kunci: BCF, STM32F103C8T6, mikrokontroler

ABSTRACT

In the industrial era 4.0, data is one of the components that must be protected. Block cipher is one of the algorithms used to secure data. This study aims to implement the block cipher four (BCF) algorithm on the microcontroller. Parameters that become benchmarks include the amount of flash and microcontroller RAM used, as well as the speed of execution of the BCF algorithm computing process. The microcontroller will run the BCF algorithm with the sequence of computing key-schedule, encryption, and decryption. Every time you start the computing process, the trigger pin on the microcontroller will send a rising signal to the oscilloscope and when it's done computing, the microcontroller trigger pin will send a falling signal to the oscilloscope. The results showed that the BCF algorithm can be implemented on the STM32F103C8T6 microcontroller. Flash and RAM used reach 22.02 Kb and 5.12 Kb. The BCF algorithm implemented on the STM32F103C8T6 microcontroller is capable of running up to 704 times faster than the NIOS II processor, 11 times faster than the AES-Engine, and 4 times slower than the BCF-Engine.

Keywords: *BCF, STM32F103C8T6, microcontroller*

1. PENDAHULUAN

Internet of Things (IoT) merupakan salah satu teknologi yang muncul pada era industri 4.0. Perangkat IoT akan bertambah pesat dari tahun ke tahun dan diperkirakan pada akhir tahun 2025 akan terdapat sekitar 7,99 miliar perangkat [1]. Perangkat IoT erat kaitannya dengan data karena tugas utama dari perangkat IoT adalah mengirim dan menerima data. Komponen data yang menjadi bagian tugas utama dari perangkat IoT tersebut menempatkan perangkat IoT dalam kondisi yang riskan, seperti terjadi kebocoran data, data yang diubah oleh pihak ketiga, dan lain sebagainya [2].

Block cipher four (BCF) merupakan algoritma enkripsi yang dibuat berdasarkan algoritma AES [3], Camelia [4], TwoFish [5], dan Khazad [6]. Algoritma ini memiliki panjang data masukan sebesar 128-bit dan panjang data kunci sebesar 128 bit, 192 bit, dan 256 bit. BCF merupakan salah satu

Naskah diterima tanggal 28 Juli 2021, disetujui tanggal 8 September 2021
*E-mail korespondensi

algoritma *block cipher* yang didesain oleh peneliti Indonesia [7]. Algoritma ini lebih aman terhadap serangan jika dibandingkan dengan algoritma AES. Algoritma AES sangat rentang terhadap serangan *correlation power analysis* (CPA) [8]. dengan menggunakan persamaan statistik [9], kunci utama AES dapat ditemukan. Algoritma BCF lebih aman karens sulitnya mencari kunci utama pada algoritma BCF. Algoritma BCF menggunakan empat buah SBOX, sedangkan AES hanya menggunakan satu SBOX. Empat SBOX yang digunakan pada algoritma BCF diimplementasikan dengan urutan acak dengan algoritma tertentu.

Penelitian sebelumnya telah mengimplementasikan algoritma BCF pada *soft-core microcontroller* NIOSII [10]. Proses implementasi dilakukan baik secara *software* maupun *hardware-accelerator*. Penelitian tersebut berfokus pada pembuatan *hardware-accelerator* untuk algoritma BCF. Hasilnya menunjukkan *BCF-Engine* mampu melakukan komputasi terhadap algoritma BCF 488-2847 kali lebih cepat dibandingkan proses komputasi BCF yang dilakukan secara *software.*

Algoritma BCF merupakan algoritma yang masih sangat jarang digunakan. Pada sektor keamanan tidak ada algoritma yang menjamin keamanan 100%. Setiap algoritma dipastikan memiliki celah. Pengembangan awal algoritma BCF ditujukan untuk digunakan pada PC. Pada penelitian ini penulis mengimplementasikan algoritma BCF pada mikrokontroler 32-bit STM32F103C8T6. Penelitian ini akan menghasilkan performa pada saat mengimplementasikan algoritma BCF pada mikrokontroler. Parameter yang menjadi tolak ukur antara lain besaran *flash* dan RAM mikrokontroler yang terpakai, serta kecepatan eksekusi proses komputasi algoritma BCF.

2. METODE PENELITIAN

2.1 Algoritma BCF

BCF menggunakan struktur Feistel [11] berbeda dengan AES yang menggunakan struktur SPN. Pada proses difusi, struktur SPN lebih cepat jika dibandingkan dengan struktur Feistel. Hal ini disebabkan struktur SPN memiliki ronde yang lebih sedikit dibandingkan dengan strukur Feistel. Proses enkripsi dan dekripsi memiliki struktur yang sama jika menggunakan struktur Feistel. Hal ini menjadi keuntungan jika algoritma BCF diimplementasikan pada *embedded system.*

Algoritma BCF memiliki dua proses yaitu *key-schedule* dan *randomizing*. Proses *key-schedule* berfungsi untuk menghitung *subkeys*. Proses *randomizing* berfungsi untuk memanipulasi data *input* menggunakan *subkeys*. Proses *randomizing* mengubah *plain text* menjadi *cipher text* (proses enkripsi) dan *cipher text* menjadi *plain text* (proses dekripsi). Banyaknya ronde yang digunakan tergantung dari panjang kunci yang digunakan. Jika panjang kunci 128-bit maka pada proses *randomizing* akan dilaksanakan sebanyak 15 ronde. Jika panjang kunci 192-bit maka pada proses *randomizing* akan dilaksanakan sebanyak 16 ronde. Jika panjang kunci 256-bit maka pada proses *randomizing* akan dilaksanakan sebanyak 18 ronde. Tiap ronde menggunakan fungsi F0, fungsi ini menggunakan *subkeys* yang berbeda untuk mengubah data input menggunakan *subkeys* yang berbeda untuk tiap ronde.

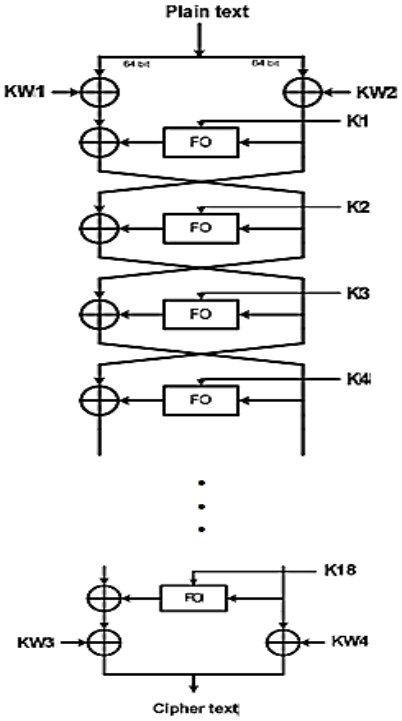
Spesifikasi dari algoritma BCF adalah sebagai berikut:

1. Panjang data *input* dan *output* adalah 128 bit.
2. Kunci utama memiliki tiga panjang data yaitu 128 bit, 192 bit, dan 256 bit.
3. Proses *key-schedule* dilakukan sebanyak 8 ronde menggunakan fungsi F0 ditambah beberapa algoritma tambahan.
4. Panjang kunci 128 bit banyaknya ronde untuk proses *randomizing* adalah 15 ronde, panjang kunci 192 bit banyaknya ronde untuk proses *randomizing* adalah 16 ronde, dan panjang kunci 256 bit banyaknya ronde untuk proses *randomizing* adalah 18 ronde.

Proses *key-schedule* dilakukan di awal sebelum dilakukan proses enkripsi maupun dekripsi. Proses *key-schedule* hanya perlu dilakukan satu kali dan perlu dilakukan komputasi ulang jika terdapat perubahan nilai kunci utama.

2.2 Enkripsi BCF

BCF memiliki *input* dengan lebar data 128 bit. Proses *randomizing* pada BCF terdiri dari 11 ronde regular, 2 ronde spesial, 2 ronde *whitening*, dan ronde final. Berikut ini merupakan diagram blok dari algoritma enkripsi BCF.



Gambar 1. Blok diagram enkripsi BCF

Fungsi F0 pada algoritma di atas digambarkan oleh persamaan berikut:

$$F0(x,k) = P(S_i(x)) \oplus k \quad (1)$$

dengan S_i merupakan operasi substitusi untuk masukan x. Operasi S dilakukan untuk tiap *byte input* $\{x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7\}$. Operasi S menggunakan SBOX seperti pada Tabel 1, Tabel 2, Tabel 3, dan Tabel 4 berikut (dalam hexadesimal):

Tabel 1. BCF SBOX 1

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0	B4	B7	67	F3	0B	94	52	68	37	3E	EF	D7	EB	8F	55	93
1	2B	A5	02	BE	21	20	98	69	8D	05	73	AF	A7	BB	DC	AC
2	54	70	95	77	C1	06	22	44	B0	C9	76	B5	2F	27	2D	32
3	72	5B	64	C0	3A	3C	49	DE	57	28	3D	0A	F4	E7	71	7A
4	D1	8E	F0	43	C8	23	7F	AA	DF	16	BA	5E	18	1A	1D	7C
5	BD	D5	85	AB	56	8C	9F	FD	4C	CA	9D	C7	ED	B9	25	CC
6	75	EE	AE	FC	4B	03	3F	A3	E3	7D	E5	D3	D4	10	E0	36
7	34	D8	3B	82	6E	89	35	01	91	79	D0	DA	5C	4	CD	FE
8	84	6B	E4	1B	6C	9B	81	CF	2C	46	A8	C5	07	26	E9	51
9	C4	D9	0F	33	4D	41	BF	61	4A	A0	A6	B1	C2	EA	66	C6
A	83	F6	E2	40	1F	5D	7B	DD	F9	B3	A4	15	8B	6A	45	09
B	E1	4E	11	50	B6	58	DB	08	CB	7E	A2	5A	F1	AD	87	FF
C	53	47	13	80	86	C3	1C	5F	A9	59	63	E6	FA	30	42	F5
D	0D	38	2A	9A	F7	90	65	CE	78	D2	9C	BC	1E	0C	17	EC
E	88	D6	6F	62	0E	6D	99	9E	8A	31	48	19	4F	00	74	FB
F	29	B8	2E	A1	39	60	96	12	97	F8	B2	24	E8	92	F2	14

Tabel 2. BCF SBOX 12

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0	40	A5	3E	EE	28	1E	51	21	60	DB	4C	59	3C	C8	8F	77
1	5A	43	2A	D2	5B	CE	A1	E9	B1	47	E3	8A	46	E1	E7	89
2	FD	BC	F9	C0	A0	F4	09	3D	52	5D	FE	A6	67	CB	EC	97
3	18	4B	6A	61	B7	C1	9E	24	4F	E5	01	03	29	08	B9	06
4	F0	A7	E6	CC	39	1D	7F	15	57	F3	82	99	70	6E	9C	58
5	D6	2C	D9	CD	A3	4D	75	48	74	2F	E2	6D	D7	12	B0	37
6	80	7E	86	79	0E	71	D0	34	0A	AC	42	94	B8	AA	56	DD
7	84	55	38	FB	1C	DC	33	DE	6C	C2	A2	D3	E4	66	AE	A4
8	9F	26	22	7	7C	F8	3B	F2	F6	1F	96	F5	85	C9	C6	AB
9	8B	45	7B	E0	2E	50	9B	D8	D1	5F	C5	65	0D	88	14	27
A	D5	20	C7	FC	44	FA	3F	62	35	A9	63	2D	49	69	19	0F
B	13	95	90	72	AD	00	31	17	0C	64	11	B4	16	53	9A	04
C	8D	CF	91	BF	D4	A8	EA	8C	C3	54	AF	93	3A	BA	1A	EB
D	EF	7D	8E	C4	6B	0B	5C	25	BD	4A	1B	68	87	B6	DF	6F
E	83	B3	BE	F7	78	81	ED	4E	BB	2B	36	B5	76	30	9D	F1
F	7A	92	E8	23	DA	CA	41	5E	10	98	FF	32	B2	73	05	02

Tabel 3. BCF SBOX 3

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0	47	EC	A1	80	49	10	D1	0B	94	50	5E	A7	23	45	B0	38
1	DE	F5	6E	D5	54	C0	8E	34	0E	BA	16	44	E2	72	AA	E5
2	26	CB	FE	60	3	BF	DD	56	57	E4	91	8C	19	65	3B	1
3	90	7B	F9	24	0C	C5	61	B7	F7	E1	37	FD	85	7E	9	2C
4	C9	AC	66	D6	40	A9	42	4	5A	6B	1C	CE	FA	AF	DB	B5
5	83	B3	A6	E7	7A	E0	CF	27	6A	EF	1E	B8	6	18	2F	63
6	B9	82	76	28	F8	ED	71	FB	70	5	C8	88	E6	4E	E3	CC
7	FC	C4	67	95	78	13	9C	2E	74	68	84	31	F2	58	AE	3D
8	DF	87	7C	2	FF	79	86	E9	2D	C2	52	5F	AD	30	8A	99
9	25	41	22	8D	1D	20	8F	97	14	77	C7	9A	F0	2B	1A	3E
A	89	9E	7	8	3A	5B	4A	CA	9F	12	C1	59	0A	55	81	A8
B	21	96	73	46	F1	C6	BD	33	0	62	1F	32	B1	7F	BC	4D
C	D2	6F	D9	D3	B2	F6	36	CD	75	DC	64	7D	6C	93	BB	A4
D	A5	F4	53	1B	35	5D	A0	A2	4B	4F	43	51	48	E8	0F	5C
E	C3	B4	DA	EE	2A	D0	39	9D	98	EA	D7	A3	AB	29	0D	EB
F	B6	17	3C	4C	F3	3F	6D	15	BE	92	8B	9B	11	D4	69	D8

Tabel 4. BCF SBOX 4

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0	2F	5A	C2	D4	9E	0B	EB	B4	6	43	D5	50	59	DF	65	4B
1	3E	99	5F	27	10	CB	42	A8	CD	C3	2C	95	E6	24	DC	B5
2	6A	AA	A6	46	C0	19	A7	73	84	FF	FA	9F	31	7F	3F	E9
3	B0	52	1B	93	3A	B3	9A	D8	FB	88	4D	D3	F3	B7	91	5
4	13	5B	D7	22	40	F4	61	75	E3	48	74	55	E5	D0	47	2D
5	11	F0	33	29	AE	9C	E4	90	68	E8	D1	EF	C5	32	6C	20
6	8B	63	D2	4F	3	D9	72	76	1D	BB	2B	1	4	AB	1E	A0
7	96	66	80	25	85	78	34	6F	E0	C9	4A	8D	7E	C6	B6	E2
8	62	0E	9D	64	82	FD	5E	71	54	CC	A4	F8	B8	94	53	30
9	35	DE	CE	2A	3C	21	FC	4E	BE	12	FE	DD	DB	6B	C4	2
A	51	AD	F7	26	8C	15	14	17	AF	0F	BF	7B	39	A3	E1	6D
B	F2	C8	CF	F9	1C	23	B2	7C	87	44	18	F1	0A	B9	79	AC
C	F6	0D	EE	98	9B	97	36	DA	1F	D6	81	CA	58	7D	8A	83
D	A9	C1	F5	BC	5D	89	77	6E	2E	B1	5C	8E	0C	28	9	1A
E	7	EA	E7	56	37	7A	41	70	57	A5	4C	67	BD	C7	60	3D
F	0	38	92	A2	69	8F	ED	BA	45	3B	A1	8	EC	16	49	86

Urutan kotak substitusi bergantung pada ronde. Tabel 5 merupakan pengaturan posisi kotak substitusi berdasarkan ronde.

Tabel 5. Pengaturan posisi kotak substitusi

Ronde	**K**	**Operasi pengacakan**
1-4	KA	$S[i] = \left((K \gg (2i + 8(r-1)))\& \, 03_x\right)$
5-8	KB	$S[i] = \left((K \gg (2i + 8(r-5)))\& \, 03_x\right)$
9-12	KA	$S[i] = \left((K \gg (2i + 8(r-9)))\& \, 03_x\right)$
13-16	KB	$S[i] = \left((K \gg (2i + 8(r-13)))\& \, 03_x\right)$
17-18	KA	$S[i] = \left((K \gg (2i + 8(r-17)))\& \, 03_x\right)$

dimana $S[i]$ menandakan nomor SBOX yang digunakan pada kotak ke-i tiap ronde di proses enkripsi dan r adalah indeks dari ronde (r = 1, 2, …, 18). Operasi perkalian matriks polinomial P dengan matriks M menggunakan persamaan berikut.

$$b = P(x) = M \,.\, x \tag{2}$$

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} = \begin{bmatrix} 03_x & 01_x & 05_x & 04_x & 0B_x & 07_x & 06_x & 08_x \\ 01_x & 03_x & 04_x & 05_x & 07_x & 0B_x & 08_x & 06_x \\ 05_x & 04_x & 03_x & 01_x & 06_x & 08_x & 0B_x & 07_x \\ 04_x & 05_x & 01_x & 03_x & 08_x & 06_x & 07_x & 0B_x \\ 0B_x & 07_x & 06_x & 08_x & 03_x & 01_x & 05_x & 04_x \\ 07_x & 0B_x & 08_x & 06_x & 01_x & 03_x & 04_x & 05_x \\ 06_x & 08_x & 0B_x & 07_x & 05_x & 04_x & 03_x & 01_x \\ 08_x & 06_x & 07_x & 0B_x & 04_x & 05_x & 01_x & 03_x \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix}$$

Variable x merupakan 64 bit masukan dan b merupakan 64 bit keluaran. Mengacu pada (1), maka

$$y = b \oplus k \tag{3}$$

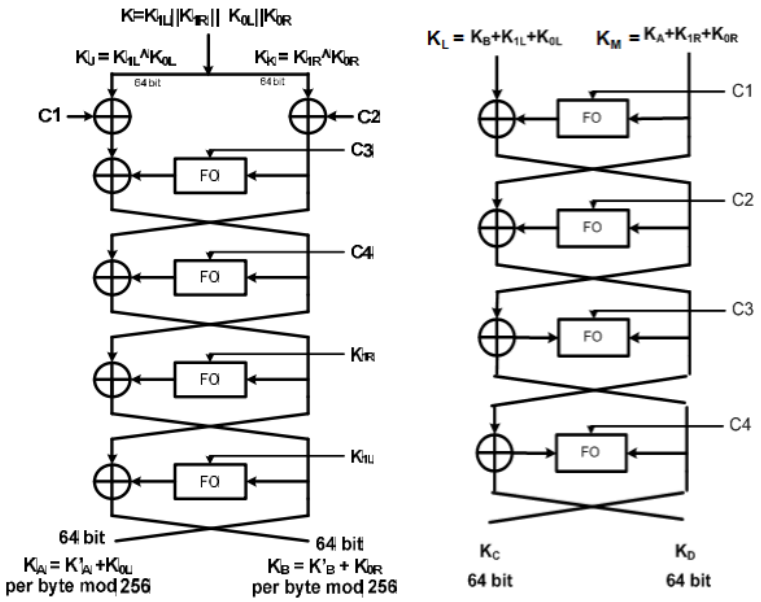
Variabel y merupakan 64 bit keluaran dari fungsi F0, b merupakan 64 bit keluaran dari fungsi perkalian polinomial P dengan matriks M, dan k merupakan 64 *subkey*.

Persamaan *irreducible polynomial* yang digunakan pada algoritma BCF adalah sebagai berikut:

$$m(x) = x^8 + x^4 + x^3 + x + 1 \quad (4)$$

2.3 *Key-schedule* BCF

Key-schedule memiliki *input* berupa kunci yang bisa berukuran 128 bit, 192 bit, ataupun 256 bit. *Key-schedule* menggunakan struktur Feistel dengan jumlah ronde sebanyak 8 untuk berbagai macam ukuran kunci. Di *key-schedule* dihasilkan *subkey* yang jumlahnya bergantung ukuran kunci *input*. Untuk ukuran kunci 128 bit menghasilkan *subkey* berjumlah 15 buah, 192 bit menghasilkan *subkey* sebanyak 16 buah, dan 256 bit menghasilkan *subkey* sebanyak 18 buah. Selain itu proses *key-schedule* juga dihasilkan kunci pengacak posisi SBOX yaitu kunci KK dan KJ untuk pengacakan kotak substitusi pada *key-schedule* serta kunci KA dan KB untuk pengacakan kunci pada enkripsi atau dekripsi. Kunci masukan berukuran 256 bit, $K = K_{1L}||K_{1R}||K_{0L}||K_{0R}$. Untuk masukan dengan ukuran kunci 192 bit maka diperoleh K_{1L} = 0, sedangkan untuk ukuran kunci 128 bit diperoleh K_{1L} = 0 dan K_{1R} = 0. BCF *key-schedule* dapat dilihat pada Gambar 2.



Gambar 2. BCF *key-schedule*

Proses kalkulasi *keyschedule* dideskripsikan pada Tabel 6.

Tabel 6. Kalkulasi *subkey*

Key	Proses kalkulasi	Key	Proses kalkulasi
KE	(KA ⋂ KB) ⊕ KC	K9	(K8 ⋂ KF) ⊕ KG
KF	(KA ⋃ KC) ⊕ KD	K10	(K9 ⋃ KC) ⊕ K8
KG	(KE ⋃ KF) ⊕ KA ⊕ KB	K11	K8 ⊕ K9 ⊕ K10
Kw1	(KE ⋂ KF) ⊕ KC	K12	$(K10 + K11)\|_{8\ bit}$
Kw2	KC ⊕ KD ⊕ KE	K13	$(K11 + K12)\|_{32\ bit}$
K1	Kw1⊕Kw2⊕0x00007a0000000000	K14	$(K13_{32l}<<<1)\|\|(K13_{32r}<<<1) \oplus K11$
K2	K1 ⊕ KC⊕KD	K15	(K12 ⋂ K14) ⊕ K13
K3	(K2⋃ 0xff00000000000000) ⊕KE	K16	(K14 ⋃ K15) ⊕ K12
K4	K3⊕KC⊕ KD	K17	(K14⋃ K15)
K5	(K4 ⋃ KF) ⊕ KG	K18	(K15⋃ K16) ⊕ K17
K6	(C1 ⋂ KE) ⊕ K5	KW3	K13 ⊕ K14
K7	(KG ⋃ K6) ⊕ KF	KW4	K15 ⊕ K16
K8	(K7 ⋂ K6) ⊕ KG		

Konstanta yang digunakan dalam proses kalkulasi *subkeys* ditampilkan pada Tabel 7.

Tabel 7. Konstanta *key*

Konstanta	Nilai
C1	0X591A4B820C6EF3D7
C2	0X9D760BE8F193205A
C3	0X492C08E51A03B758
C4	0XF750A2B38E41C96D

Pengacakan posisi kotak substitusi ditentukan oleh kunci KJ, KK untuk ronde 1-4 dan kunci KL, KM untuk ronde 5-8 seperti ditunjukkan pada Tabel 8.

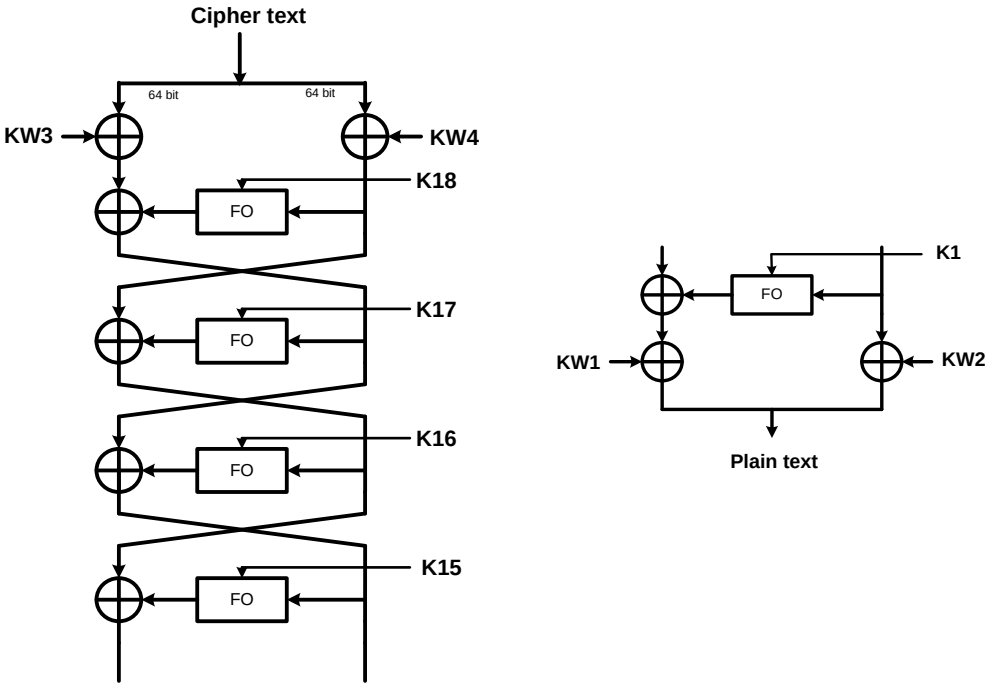
Tabel 8. Pengaturan posisi kotak substitusi pada tiap ronde pada proses *key-schedule*

Ronde	K	Operasi pengacakan
1-2	KJ	$S[i] = \bigoplus_{n=0}^{3} \left((K \gg (2n+8i)) \& 03_x\right)$
3-4	KK	$S[i] = \bigoplus_{n=0}^{3} \left((K \gg (2n+8i)) \& 03_x\right)$
5-6	KL	$S[i] = \left((K \gg 8i) \& 03_x\right)$
7-8	KM	$S[i] = \left((K \gg 8i) \& 03_x\right)$

dengan $S[i]$ menandakan nomor kotak substitusi yang digunakan pada kotak ke-i tiap ronde di *key-schedule*.

2.4 Dekripsi BCF

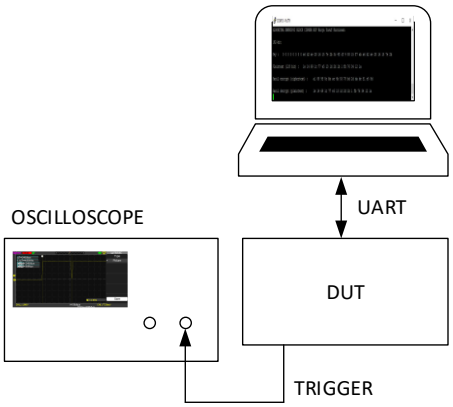
Mengubah *cipher text* menjadi *plain text* (proses dekripsi) dapat dilakukan dengan proses yang sama dengan mengubah *plain text* menjadi *cipher text* (proses enkripsi). Perbedaan terletak pada urutan *subkey* yang digunakan untuk tiap ronde. Gambar 3 merupakan blok diagram dari proses dekripsi BCF.



Gambar 3 Blok diagram dekripsi BCF

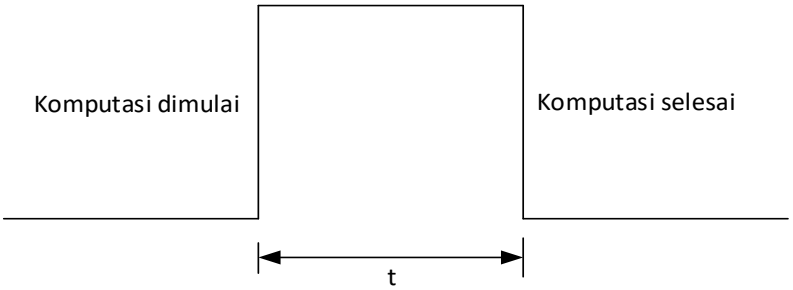
2.5 Perancangan Sistem Mikrokontroler

Algoritma BCF diterjemahkan menjadi sebuah kode menggunakan Bahasa C. Kode yang dibuat dalam penelitian ini benar-benar menerjemahkan algoritma BCF tanpa melakukan optimasi, baik optimasi kecepatan eksekusi komputasi maupun optimasi penggunaan memori, baik flash maupun RAM mikrokontroler. Blok diagram rangkaian yang diimplementasikan algoritma BCF ditunjukkan oleh Gambar 4.



Gambar 4. Blok diagram pengujian

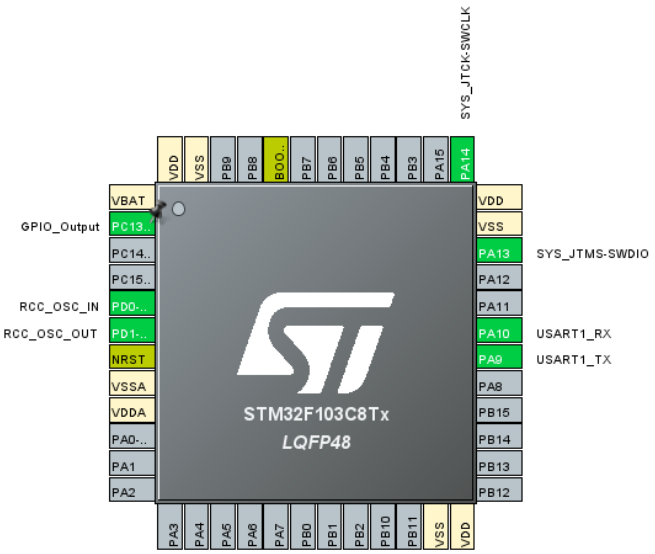
Mikrokontroler akan menjalankan algoritma BCF dengan urutan komputasi *key-schedule*, enkripsi, lalu dekripsi. Seperti diilustrasikan pada Gambar 5, setiap kali memulai proses komputasi pin *trigger* pada mikrokontroler akan mengirimkan sinyal *rising* ke osiloskop dan pada saat selesai melakukan komputasi maka pin *trigger* mikrokontroler akan mengirimkan sinyal *falling* ke osiloskop.



Gambar 5. Sinyal *trigger*

Waktu komputasi didapatkan dengan menghitung lebar sinyal (t). Metode ini dipilih karena tidak menggangu waktu komputasi algoritma BCF. Kode program dibuat dua, yaitu kode yang dijalankan di PC dan mikrokontroler. Mikrokontroler akan mengirimkan data ke PC menggunakan komunikasi UART lalu ditampilkan pada *serial monitor*. Langkah ini berfungsi untuk memverifikasi apakah algoritma yang dijalankan pada mikrokontroler berjalan sebagaimana mestinya atau tidak, dengan cara membandingkan hasil komputasi BCF pada PC dan mikrokontroler.

Tipe mikrokontroler yang akan digunakan adalah STM32F103C8T6. Mikrokontroler ini menggunakan arsitektur ARM-Cortex M3 dengan konfigurasi pin seperti ditunjukkan Gambar 6. *Core* mikrokontroler ini mampu berjalan dengan kecepatan 72 MHz dengan memanfaatkan internal PLL. UART1 digunakan untuk berkomunikasi dengan PC dengan *baudrate* yang digunakan sebesar 115200 bps. Pin PC13 digunakan sebagai *trigger* yang disambungkan dengan osiloskop.



Gambar 6. Konfigurasi pin mikrokontroler

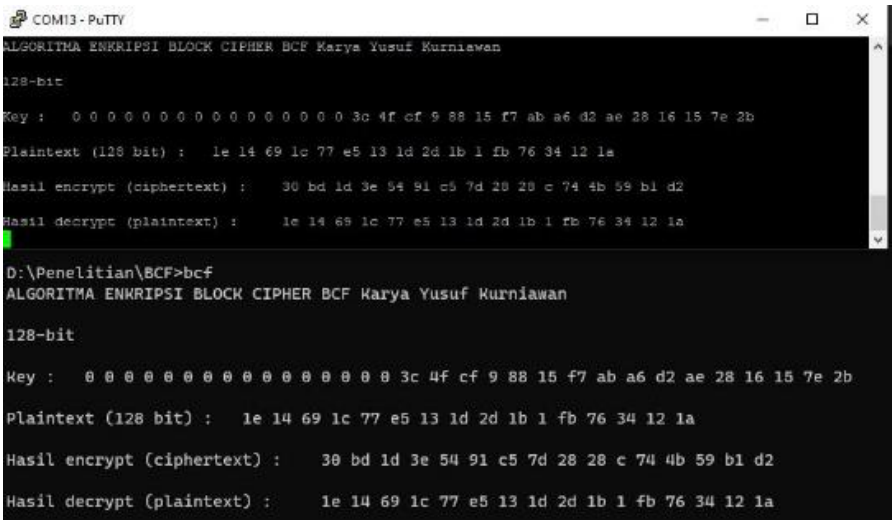
3. HASIL DAN PEMBAHASAN

Pengujian dilakukan dalam dua tahap yaitu tahap verifikasi dan tahap pengujian. Tahap verifikasi bertujuan memastikan hasil komputasi yang dilakukan pada mikrokontroler dan PC menghasilkan nilai yang sama. Tahap pengujian bertujuan untuk mengetahui seberapa besar memori yang dikonsumsi (baik RAM maupun *flash*) dan seberapa cepat mikrokontroler STM32F103C8T dapat melakukan komputasi algoritma BCF. Tahap verifikasi dilakukan tiga kali pengujian, yaitu pengujian pada saat menggunakan kunci sepanjang 128 bit, 192 bit, dan 256 bit. Matriks pengujian ditunjukkan pada Tabel 9.

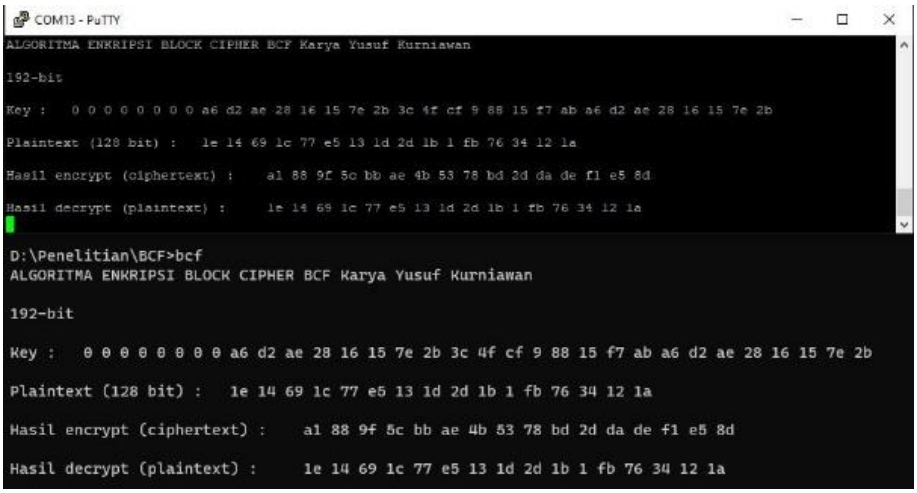
Tabel 9. Matrik pengujian

Nama	Nilai (HEX)
Plain Text	1E14691C77E5131D2D1B01FB7634121A
KEY 128-bit	3C4FCF098815F7ABA6D2AE2816157E2B
KEY 192-bit	6D2AE2816157E2B3C4FCF098815F7ABA6D2AE2816157E2B
KEY 256-bit	3C4FCF098815F7ABA6D2AE2816157E2B3C4FCF098815F7ABA6D2AE2816157E2B

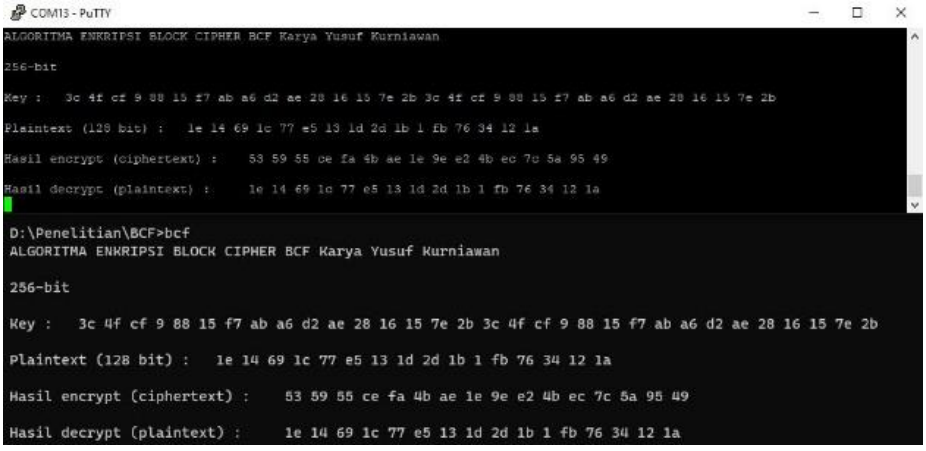
Gambar 7, Gambar 8, dan Gambar 9 menunjukkan perbandingan hasil komputasi BCF yang dilakukan oleh mikrokontroler dan PC.



Gambar 7. *Test vector* BCF 128 bit



Gambar 8. *Test vector* BCF 192 bit



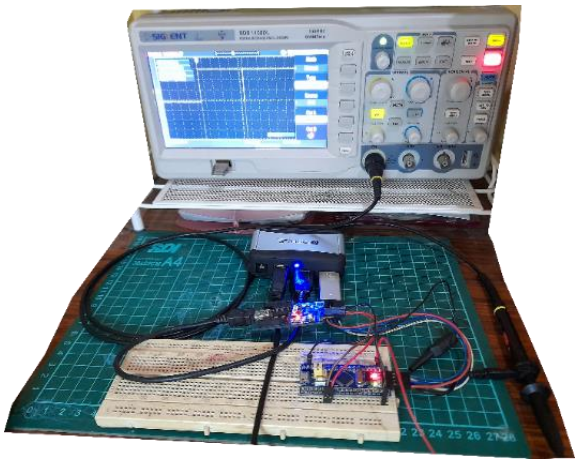
Gambar 9. *Test vector* BCF 256 bit

Gambar tersebut menunjukkan bahwa komputasi algoritma BCF yang dilakukan pada mikrokontroler dan PC menghasilkan nilai yang sama. Tabel 10 menunjukkan besaran memori yang terpakai pada saat mengimplementasikan algoritma BCF. Hasilnya menunjukkan bahwa *flash* dan RAM terpakai sebesar 34% dan 27% dari ukuran maksimalnya. IDE yang digunakan adalah STM32CubeIDE dengan menggunakan HAL Library.

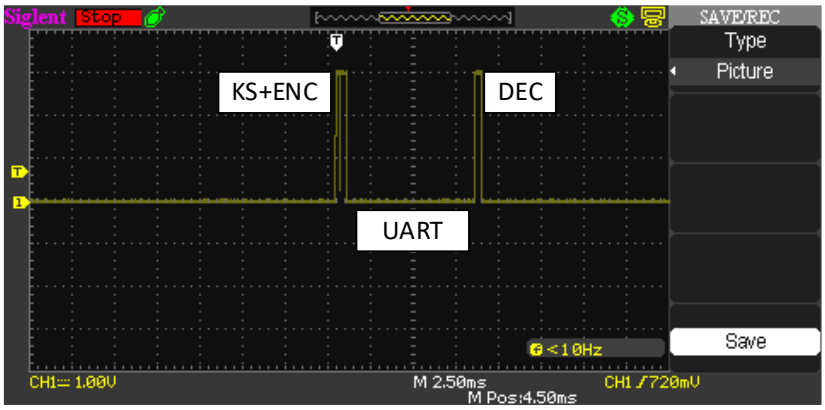
Tabel 10. Penggunaan memori

Nama Memori	Ukuran Memori	Memori Terpakai
Flash	64 Kb	22,02 Kb
RAM	20 Kb	5,12 Kb

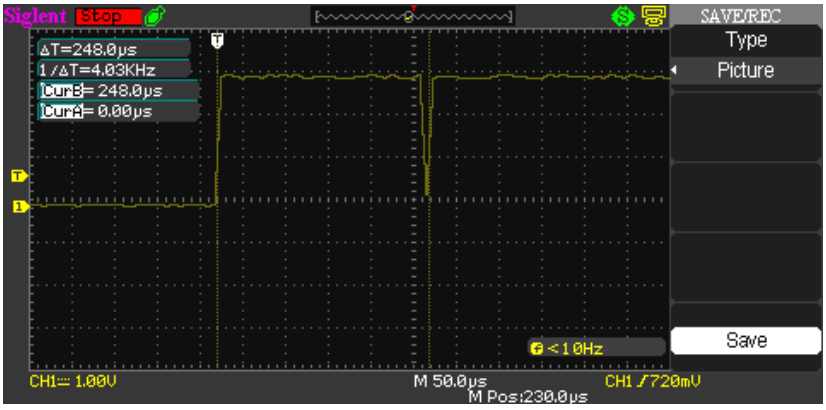
Pengujian kecepatan dilakukan menggunakan konfigurasi seperti pada Gambar 10. Tipe osiloskop yang digunakan adalah SDS-1102DL dengan *bandwith* sebesar 100 MHz dan kecepatan *sampling* sebesar 500 MS/s. Mikrokontroler STM32F103C8T6 menggunakan *crystal* eksternal sebesar 8 MHz dan kecepatan *clock* yang digunakan diperbesar sampai dengan 72 MHz dengan memanfaatkan internal PLL. Pin 13 pada PORTC digunakan sebagai sinyal *trigger*. Pin ini dikonfigurasikan dengan kecepatan *high*, sehingga *slope* yang dihasilkan menjadi sangat kecil dan tidak akan terlalu mempengaruhi hasil pengukuran. Gambar 11 dan Gambar 12 menunjukkan hasil akuisisi data yang dihasilkan oleh osiloskop.



Gambar 10. Konfigurasi pengujian



Gambar 11. Sinyal yang diakuisisi osiloskop



Gambar 12. Proses pengukuran lebar pulsa

Gambar 11 menunjukkan terdapat dua buah sinyal pulsa. Sinyal pulsa tersebut terdiri dari sinyal pada saat melakukan proses *key-schedule* + enkripsi dan sinyal pada saat melakukan proses dekripsi. Diantara proses enkripsi dan dekripsi terdapat *delay* yang cukup besar. Hal ini dikarenakan setelah melakukan proses enkripsi, mikrokontroler mengirimkan data *cipher* ke PC terlebih dahulu.

Analisis terhadap sinyal dilakukan dengan mengukur lebar pulsa. Proses pengukuran ditunjukkan oleh Gambar 12. Hasil pengukuran kecepatan eksekusi komputasi BCF ditunjukkan oleh Tabel 11.

Tabel 11. Komparasi kecepatan komputasi

Nama Proses	Waktu Komputasi (uS)		
	128 bit	192 bit	256 bit
Key-schedule	244	248	248
Enkripsi	360	380	428
Dekripsi	350	370	420
Total	954	998	1096

Tabel 12. Perbandingan kecepatan dengan mikrokontroler lain

Mikrokontroler	Waktu Eksekusi (µS)					
	256 bit		192 bit		128 bit	
	Software	**BCF *Engine***	***Software***	**BCF Engine**	***Software***	**BCF *Engine***
STM32103C8T6	1096	-	998	-	954	-
NIOS II *Fast*	141664	290	135977	255	127704	239
NIOS II *Standart*	172815	273	159609	261	153015	253
NIOS II *Economy*	760116	267	701536	267	672323	267

Tabel 11 menunjukkan bahwa proses komputasi enkripsi selalu menghabiskan waktu lebih lama jika dibandingkan dengan waktu komputasi proses yang lain. Semakin besar kunci yang digunakan maka waktu yang dibutuhkan semakin lama.

Tabel 12 membandingkan dengan hasil dari penelitian sebelumnya [10]. Hasil perbandingan menunjukkan mikrokontroler STM32F103C8T6 lebih cepat melakukan komputasi algoritma BCF pada implementasi secara *software*. Mikrokontroler STM32F103C8T6 129-704 kali lebih cepat, tergantung pada panjang kunci yang digunakan. Mikrokontroler yang digunakan pada penelitian sebelumnya menggunakan *clock* sebesar 50 MHz, sedangkan pada penelitian ini menggunakan *clock* sebesar 72 MHz. Mikrokontroler NIOS II merupakan *soft-core* mikrokontroler sedangkan mikrokontroler STM32F103C8T6 merupakan *hard-core* mikrokontroler. Tabel 13 menunjukan perbandingan kecepatan eksekusi algoritma AES dan BCF yang diimplementasikan pada mikrokontroler. Data pada Tabel 13 menunjukkan hasil kecepatan eksekusi pada saat menggunakan *hardware accelerator*, kecuali data yang merepresentasikan hasil dari penelitian ini. Dari hasil perbandingan dapat dilihat bahwa STM32F103C8T6 10 kali lebih cepat dibandingkan dengan AES-*Engine* dan 4 kali lebih lambat jika dibandingkan dengan BCF-*Engine*.

Tabel 13. Komparasi dengan *hardware accelerator*

Desain	Algoritma	Waktu Eksekusi (uS)
Penelitian [2]	AES	10414
NIOS II + BCF-Engine	BCF	239
Penelitian ini	BCF	954

4. KESIMPULAN

Algoritma BCF dapat diimpelentasikan pada mikrokontroler STM32F103C8T6. *Flash* dan RAM yang digunakan mencapai 22,02 Kb dan 5,12 Kb. Bagi mikrokontroler yang memiliki kapasitas memori yang cukup besar, hal tersebut tidak menjadi masalah. Namun jika algoritma BCF ini diimplementasikan pada mikrokontroler dengan kapasitas memori yang kecil, maka akan menjadi masala. Hal ini karena algoritma BCF ini hanya berfungsi untuk mengamankan data, belum termasuk kode program utama. Ukuran memori yang terpakai bergantung kepada *compiler* dan *library* yang digunakan. Algoritma BCF yang diimplementasikan pada mikrokontroler STM32F103C8T6 mampu berjalan cepat jika dibandingkan dengan prosesor NIOS II. Mikrokontroler STM32F103C8T6 mampu berjalan sampai dengan 704 kali lebih cepat. STM32F103C8T6 hanya lebih lambat 4 kali jika dibandingkan dengan BCF-*Engine*. Bahkan dengan AES-*Engine*, STM32F103C8T6 mampu berjalan 11 kali lebih cepat. Agar algoritma BCF ini mampu diimplementasikan di berbagai arsitektur mikrokotroler, diharapkan kedepannya dibuat implementasi secara *software* yang berfokus untuk

mengoptimalisasi penggunaan memori atau kecepatan. Selain itu, untuk penelitian kedepannya dapat dapat digunakan LL Library.

UCAPAN TERIMA KASIH

Tim peneliti mengucapkan terimakasih kepada Ramdoo Electronics yang telah memfasilitasi peralatan dalam melakukan penelitian ini.

REFERENSI

[1] A. H. Alavi, P. Jiao, W. G. Buttlar, and N. Lajnef, "Internet of Things-enabled smart cities: State-of-the-art and future trends," *Measurement*, vol. 129, pp. 589-606, 2018.
[2] A. Sideris, T. Sanida, and M. Dasygenis, "Hardware Acceleration of the AES Algorithm using Nios-II Processor," *2019 Panhellenic Conference on Electronics & Telecommunications (PACET)*, Volos, Greece, 2019, pp. 1-5.
[3] AES (*Advanced Encryption Standard*), FIPS PUB 197, Federal Information Processing Standards Publication, 2001.
[4] K. Aoki, T. Ichikawa, M. Kanda, M. Matsui, S. Moriai, J. Nakajima, and T. Tokita, "Specification of Camellia-a 128-bit block cipher," *Specification Version 2*, 2000.
[5] B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, and N. Ferguson, *The Twofish encryption algorithm: a 128-bit block cipher*. John Wiley & Sons, Inc., 1999.
[6] P. Barreto and V. Rijmen, "The Khazad legacy-level block cipher," *Primitive submitted to NESSIE*, vol. 97, no. 106, 2000.
[7] K. Yusuf, A. Fauzan, and L. Muhammad, *Desain dan analisis sandi BCF*. Bandung: Tidak diterbitkan. 2015.
[8] S. D. Putra, M. Yudhiprawira, Y. Kurniawan, S. Sutikno, and A. S. Ahmad, "Security Analysis of BC3 Algorithm for differential Power Analysis Attack," in *International Syposium on Elecronics and Smart Devices (ISESD)*, Yogyakarta, 2017, pp. 341-345.
[9] P. C. Kocher, "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems," in *Annual International Cryptology Conference*, Berlin, 1996, pp. 104-113.
[10] Y. Kurniawan and M. A. Rizqulloh, "Block cipher four implementation on field programmable gate array," *Communications in Science and Technology*, vol. 5, no. 2, pp. 53-64, 2020.
[11] W. Stalling, *Cryptography and Network Security Principles and Practices (4th ed)*, New Jersey: Prentice Hall, 2005.