Lightweight Cryptography and the Internet of Things

Mark Bell Edinburgh Napier University, Edinburgh, UK.

Abstract

In recent years, there has been a substantial growth in internet connected devices that form the Internet of Things. Within the Internet of Things, resource restricted devices such as RFID chips and wireless sensor networks are connected to the cloud and require encryption to protect the data stored and transmitted by these devices. Lightweight cryptography balances the limited resource available from small devices with the to ensure protection of data handled by these devices.

Keywords: Lightweight cryptography, Skinny, Speck, Simon, FELICS

1. Introduction

This paper has been produced to evaluate existing lightweight cryptography methods suitable for Internet of Things (IoT) devices [2]. As the world becomes increasingly networked, hostile actors are seeking to capitalise on this and are consistently looking for new ways to exploit connected devices. This paper consists of two parts:

1. A literature review examining existing research into the internet of things and lightweight cryptography.
2. Testing and implementation of lightweight encryption algorithms.

2. Literature Review

The IoT refers to internet connected devices. Andrea et al [12] describe the IoT as networked system connecting a variety of devices such as televisions with a WiFi connection, internet connected heating systems, and smart speakers which are connected to the internet. IoT devices are used extensively across many industries and form a key part of the IT infrastructure of

an organisation. Three key technologies that enable the Internet of things are:

1. Radio Frequency ID (RFID): RFID uses electromagnetic fields, enabling radio chips to communicate data wirelessly.
2. Wireless sensor networks (WSN): WSN consist of a collection of geographically dispersed which are used for monitoring factors such as temperature, light and and pressure.
3. Cloud computing: Cloud computing in the context of IoT devices allows for data collected from sensors to stored and intelligently monitored.

[1]

[1] Lightweight cryptography is required to ensure that end to end communications remain efficient and that lower resourced devices are able to be protected [4].

Industry standard encryption and hashing algorithms such as AES and SHA work well in a traditional computing environment but demand too much resource for IoT from devices. Lightweight cryptography focuses on encryption algorithms for IoT technologies such as embedded systems and RFID and sensor networks, that are constrained in resource such as processing power, physical space on disk, and volatile memory. A trade off must be made between the security offered through traditional encryption, and the constraints put on the algorithm by IoT device resources. This literature review will evaluate IoT devices, security issues, and constraints. Existing encryption methods will then be examined prior to the implementation phase of this paper.

2.1. Background

founder of the 'Auto-ID Center' at MIT, during a presentation he was delivering to Proctor and Gamble. Ashton used the term to describe RFID which was the subject of his presentation. The first internet connected device was a soft drinks dispenser at Carnegie Mellon University in the 1980s. Computer programmer at the time connected the machine to a network so that they could check the status of the machine and determine whether there would be a cold drink available if they chose to visit the machine [1]. Since this initial introduction, the scope of IoT devices has grown exponentially and this is unlikely to change going forward.

2.2. Legislation

In 2018, the UK Government Department for Culture, Media, and Sport (DCMS), published a white paper called 'secure by design' [7] which addresses substantial growth in an unregulated industry of IoT devices. The paper addresses the fact that a large proportion of IoT devices are sold with no security considerations and placed the onus to secure the device on the consumer. The DCMS reported that this had two primary risks:

1. Threat to consumer security which is being undermined by vulnerable devices
2. threat to the UK economy as the likelihood of a largescale cyber-attack increases from a large volume of insecure devices.

The publication of [7] included guidelines for developers of IoT devices to build security into IoT devices instead of relying on the consumer to implement measures after purchasing the device. This report preceded a wider campaign to introduce legislation mandating that IoT devices are 'secure by design'. In May 2019, Margot James of the DCMS announced [8] that a 'consultation on regulatory next steps' was launching with the intent of mandating an IoT code of practise for new devices. While this is a positive first step for new devices, it does not address the issue of existing unprotected IoT devices.

2.3. IoT Security Considerations

The security of an IoT device can be classified according to the confidentiality integrity and availability (CIA) triad. The CIA triad is a model which is designed to guide an organisation with classifying the protection of assets. The three components of the component of the CIA triad are defined as such:

- **Confidentiality:** maintains the privacy of the asset. Within an organisation, this may refer to company sensitive information which can only be viewed by a small number of authorised employees. To maintain privacy in this case, access should not be granted to unauthorised personnel.

- **Integrity:** ensures the content of the asset is not compromised either through loss, damage, or change.

- **Availability:** ensures the asset remains available. An example could be a share drive within an organisation containing company critical information which becomes unavailable due to network failure.

In 6, a CIA model tailored to IoT devices was created which considered which assets of an IoT device would need protecting for each component of the CIA triad.

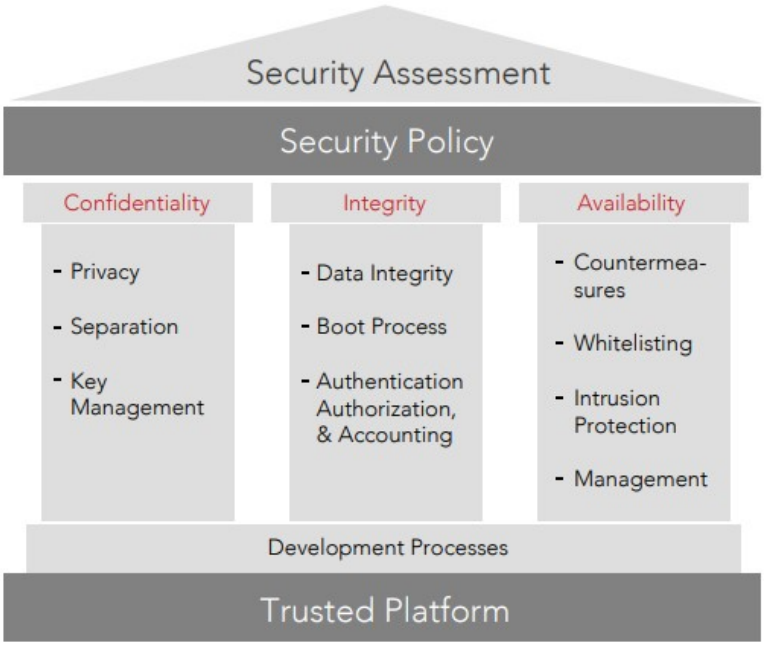

Figure 1: IoT CIA Triad

Increasingly, IoT devices are being targeted as an attack vector for hostile actors to gain entry to an organisation or are recruited to provide extra strength to botnets such as the Mirai botnet. The Mirai botnet exploits devices that use an 'Arc Processor' and have embedded unchanged default passwords.

2.4. lightweight attacks

IoT architecture can be broken down into the application layer, communication layer, and the physical layer. Figure 2 shows an example of services operating at each of these layers.

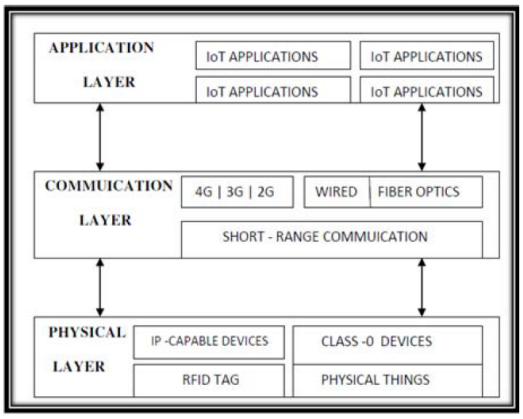

Figure 2: IoT architecture [6]

There are several known attacks targeting lightweight encryption. In [12] IOT attacks are categorised according to type.

Physical Attacks	**Network Attacks**	**Software Attacks**	**Encryption Attacks**
Node Tampering	Traffic Analysis Attacks	Virus and Worms	Side Chanel Attacks
RF Interference	RFID Spoofing		Cryptanalysis Attacks: a)Ciphertext Only Attack b)Known Plaintext Attack c)Chosen Plaintext or Ciphertext Attack
Node Jamming	RFID Cloning	Spyware and Adware	
Malicious Node Injection	RFID Unauthorised Access		
Physical Damage	Sinkhole Attack	Trojan Horse	
Social Engineering	Man In the Middle Attack		
Sleep Deprivation Attack	Denial of Service	Malicious scripts	Man In the Middle Attack
Malicious Code Injection on the Node	Routing Information Attacks	Denial of Service	
	Sybil Attack		

Figure 3: IoT attack categories [12]

Figure 3 shows some examples of encryption based attacks that have been observed targeting internet of things devices.

Side channel attacks: A side channel attack utilises techniques to exploit aspects of the encryption function such as timing, power utilised, electromagnetic analysis with the aim of recovering some information about encryption and decryption techniques. This information would be used to make the computational resource required less onerous and therefore more likely to succeed. Going forward,lightweight encryption will be written to ensure that

3. Lightweight encryption methods

3.1. Lightweight limitations

In [13], Beierle et al propose that lightweight encryption can be further categorised into two types:

Ultra lightweight cryptography: deals with encryption algorithms fulfilling a unique purpose while satisfying specific and narrow constraints.

Ubiquitous lightweight cryptography: more versatile algorithms both in terms of functionality and in terms of implementation trade-offs.

IoT devices are constrained by several factors which can be categorised based on whether encryption is implemented in hardware, or in software [13]. Hardware limitations can be described as such:

1 Gate equivalents: the memory consumption and implementation size are combined to total the gate area, which is measured in gate equivalents.

2 Throughput: throughput is measured in bits or bytes per second.

3 Latency: measured in seconds and corresponds to the time taken to obtain the output of the computation once input has been entered.

4 Power consumption: measured in Watts and is the power required to function. In the context of RFID this is particularly relevant since RFID devices depend on a passive power supply obtained from radio waves as it has no attached power supply of it's own. RFID therefore would require that the encryption algorithm favours a very low power consumption.

An example of a An encryption algorithm must balance these four factors against the strength of the encryption. Lightweight cryptography can be implemented in software as well as hardware. Lightweight cryptography in software is often seen in micro controllers. Metrics that should be considered in a software environment are listed below:

Software considerations:

1 RAM consumption: This corresponds to the amount of data written to memory for each evaluation of the function

2 Size on disk: Typically, lightweight devices are very small and will have limited spare physical storage

3 throughput: As with hardware considerations, throughput should be factored into the encryption implementation

3.2. ISO/IEC 29192 – LIGHTWEIGHT CRYPTOGRAPHY STANDARDS

ISO/IEC 29192 is comprised of six parts. NIST provided an overview of each part and this contains [11]:

Part 1

Minimum key size 80 bits [9,3]

Part 2 Block Cipher[10].

PRESENT, CLEFIA, SIMON, SPECK, SKINNY

Part 3 STREAM CIPHERS

Enocoro Trivium

Part 4 ASYMMETRIC ENCRYPTION

ELLI(RFID)

Part 5 Hash Functions

PHOTON, SPONGENT

Part 6 MESSAGE AUTHENTICATION CODES

Chaskey (MAC). Chaskey. Chaskey is a light-weight MAC function.

While the list above provides some examples of existing lightweight cryptography there is a wide range of application, and even security provided within today's lightweight encryption methods.There have been calls from several organisation's such as NIST to standardise lightweight cryptography and concerns have been raised that lightweight encryption works with too 'narrow a margin of safety'. This is referring to the amount of time taken to brute force a given length key taking into consideration the amount of computing power being used. With computing power increasing year on year, ciphers become inherently weaker with time. With lightweight cryptography trading security for efficiency, the existing encryption algorithms will be considered vulnerable more quickly than traditional encryption algorithms.

As a counter to these concerns, there is ongoing evolution of chip technology which is leading to gate sizes in chips from 14nm, 10nm, and 7nm. Limitations in gate sizes previously have come from thermal and power constraints where smaller chips had no way of dissipating excess heat causing overheating issues [16, 17].

3.3. SKINNY lightweight encryption

SKINNY lightweight encryption described in [18] was created to compete with SIMON encryption algorithm. SIMON and SPECK belong to the same encryption family and were created by the NSA. SKINNY was built with extra protection against side channel attacks.

SKINNY is also presented as 'MANTIS', which is a variant of SKINNY, and is written with a 'tweakable' block cipher. A tweakable block cipher contributes to data security, but previous iterations existed in PRINCE, but came with a high memory overhead unsuitable for lightweight devices. MANTIS solves this issue with a lightweight version of previously existing, well known and well tested code, by relying on a high number of Sboxes both in single key setting and in a setting where the attacker can control the difference in a tweak.

SKINNY is available in 64 bit and 128 bit block versions and follows the TWEAKEY framework from [19] and takes a tweakey input rather than a key or a key pair/ tweak. No reporting was found to suggest that a succesful attack has been mounted against SKINNY encryption. SIMON and SPECK have been subjected to differential analysis attacks, and AES has been subjected to key recovery attacks on variants up to 10 rounds.

SKINNY will be evaluated against other encryption algorithms in the implementation phase of this paper to assess performance.

4. Tools

The Fair Evaluation of Lightweight Cryptographic Systems (FELICS) bench-marking framework was created by CryptoLux. CryptoLux are a cryptography research group comprised of Security and Trust interdisciplinary center (SnT) and the Computer Science and Communications (CSC) research unit of the University of Luxembourg.

FELICS was developed by CryptoLux to introduce a 'fair' way of testing encryption algorithms. They note that existing research is conducted against substantially different baselines, making it difficult to draw meaningful conclusions drawn from data from different platforms.

FELICS provide definitions for results that are returned using the FELICS platform. These are summarised below [15].

1. key sizes, Block sizes are in bits [b].
2. Code size 'Code' and RAM allocation 'RAM' are in bytes [B].
3. code execution time 'Time' is in cycles [cyc.].
4. the Security level (Sec.) is defined as the ratio of the number of rounds broken in a single key setting to the total number of rounds.
5. For ciphers which have not been found to have been successfully attacked, the security level is set to -1.
6. Results for assembly implementations are displayed in italic.
7. Cipher-r denotes the 'Cipher' with 'r' rounds instead of the default number of rounds.

5. metrics

FELICS allows for bench-marking the following software focused metrics:

1. RAM overheads
2. size on disk
3. cipher execution time

6. test plan

For the purpose of this implementation, the scope testing will be limited to block ciphers only. lightweight encryption algorithm SKINNY-128-128 will be compared to AES-128-128 as a similar key size. Speck-64-96 and Simon-64-96 will also be bench-marked against SKINNY, since SKINNY was created to offer direct competition to this family of algorithms.

7. Implementation

7.1. encryption key schedule

This code excerpt shows an excerpt of encrypted keys used by SKINNY 128

```
#include <stdint.h>

#include "constants.h"


/*
 *
 * Cipher constants
 *
 */
/* Replace with the cipher constants definition */
SBOX_BYTE SBOX[256] = {
        0x65 , 0x4c , 0x6a , 0x42 , 0x4b , 0x63 , 0x43 , 0x6b ,
            ↪  0x55 , 0x75 , 0x5a , 0x7a , 0x53 , 0x73 , 0x5b ,
            ↪  0x7b ,
        0x35 , 0x8c , 0x3a , 0x81 , 0x89 , 0x33 , 0x80 , 0x3b ,
            ↪  0x95 , 0x25 , 0x98 , 0x2a , 0x90 , 0x23 , 0x99 ,
            ↪  0x2b ,
        0xe5 , 0xcc , 0xe8 , 0xc1 , 0xc9 , 0xe0 , 0xc0 , 0xe9 ,
            ↪  0xd5 , 0xf5 , 0xd8 , 0xf8 , 0xd0 , 0xf0 , 0xd9 ,
            ↪  0xf9 ,
        0xa5 , 0x1c , 0xa8 , 0x12 , 0x1b , 0xa0 , 0x13 , 0xa9 ,
            ↪  0x05 , 0xb5 , 0x0a , 0xb8 , 0x03 , 0xb0 , 0x0b ,
            ↪  0xb9 ,
        0x32 , 0x88 , 0x3c , 0x85 , 0x8d , 0x34 , 0x84 , 0x3d ,
            ↪  0x91 , 0x22 , 0x9c , 0x2c , 0x94 , 0x24 , 0x9d ,
            ↪  0x2d ,
        0x62 , 0x4a , 0x6c , 0x45 , 0x4d , 0x64 , 0x44 , 0x6d ,
            ↪  0x52 , 0x72 , 0x5c , 0x7c , 0x54 , 0x74 , 0x5d ,
            ↪  0x7d ,
        0xa1 , 0x1a , 0xac , 0x15 , 0x1d , 0xa4 , 0x14 , 0xad ,
            ↪  0x02 , 0xb1 , 0x0c , 0xbc , 0x04 , 0xb4 , 0x0d ,
```

```
        ↪ 0xbd ,
    0xe1 , 0xc8 , 0xec , 0xc5 , 0xcd , 0xe4 , 0xc4 , 0xed ,
        ↪ 0xd1 , 0xf1 , 0xdc , 0xfc , 0xd4 , 0xf4 , 0xdd ,
        ↪ 0xfd ,
    0x36 , 0x8e , 0x38 , 0x82 , 0x8b , 0x30 , 0x83 , 0x39 ,
        ↪ 0x96 , 0x26 , 0x9a , 0x28 , 0x93 , 0x20 , 0x9b ,
        ↪ 0x29 ,
    0x66 , 0x4e , 0x68 , 0x41 , 0x49 , 0x60 , 0x40 , 0x69 ,
        ↪ 0x56 , 0x76 , 0x58 , 0x78 , 0x50 , 0x70 , 0x59 ,
        ↪ 0x79 ,
    0xa6 , 0x1e , 0xaa , 0x11 , 0x19 , 0xa3 , 0x10 , 0xab ,
        ↪ 0x06 , 0xb6 , 0x08 , 0xba , 0x00 , 0xb3 , 0x09 ,
        ↪ 0xbb ,
    0xe6 , 0xce , 0xea , 0xc2 , 0xcb , 0xe3 , 0xc3 , 0xeb ,
        ↪ 0xd6 , 0xf6 , 0xda , 0xfa , 0xd3 , 0xf3 , 0xdb ,
        ↪ 0xfb ,
    0x31 , 0x8a , 0x3e , 0x86 , 0x8f , 0x37 , 0x87 , 0x3f ,
        ↪ 0x92 , 0x21 , 0x9e , 0x2e , 0x97 , 0x27 , 0x9f ,
        ↪ 0x2f ,
    0x61 , 0x48 , 0x6e , 0x46 , 0x4f , 0x67 , 0x47 , 0x6f ,
        ↪ 0x51 , 0x71 , 0x5e , 0x7e , 0x57 , 0x77 , 0x5f ,
        ↪ 0x7f ,
    0xa2 , 0x18 , 0xae , 0x16 , 0x1f , 0xa7 , 0x17 , 0xaf ,
        ↪ 0x01 , 0xb2 , 0x0e , 0xbe , 0x07 , 0xb7 , 0x0f ,
        ↪ 0xbf ,
    0xe2 , 0xca , 0xee , 0xc6 , 0xcf , 0xe7 , 0xc7 , 0xef ,
        ↪ 0xd2 , 0xf2 , 0xde , 0xfe , 0xd7 , 0xf7 , 0xdf ,
        ↪ 0xff};

SBOX_BYTE INV_SBOX[256] = {
    0xac , 0xe8 , 0x68 , 0x3c , 0x6c , 0x38 , 0xa8 , 0xec ,
        ↪ 0xaa , 0xae , 0x3a , 0x3e , 0x6a , 0x6e , 0xea ,
        ↪ 0xee ,
    0xa6 , 0xa3 , 0x33 , 0x36 , 0x66 , 0x63 , 0xe3 , 0xe6 ,
        ↪ 0xe1 , 0xa4 , 0x61 , 0x34 , 0x31 , 0x64 , 0xa1 ,
        ↪ 0xe4 ,
    0x8d , 0xc9 , 0x49 , 0x1d , 0x4d , 0x19 , 0x89 , 0xcd ,
        ↪ 0x8b , 0x8f , 0x1b , 0x1f , 0x4b , 0x4f , 0xcb ,
```

```
    ↪ 0xcf ,
0x85 , 0xc0 , 0x40 , 0x15 , 0x45 , 0x10 , 0x80 , 0xc5 ,
    ↪ 0x82 , 0x87 , 0x12 , 0x17 , 0x42 , 0x47 , 0xc2 ,
    ↪ 0xc7 ,
0x96 , 0x93 , 0x03 , 0x06 , 0x56 , 0x53 , 0xd3 , 0xd6 ,
    ↪ 0xd1 , 0x94 , 0x51 , 0x04 , 0x01 , 0x54 , 0x91 ,
    ↪ 0xd4 ,
0x9c , 0xd8 , 0x58 , 0x0c , 0x5c , 0x08 , 0x98 , 0xdc ,
    ↪ 0x9a , 0x9e , 0x0a , 0x0e , 0x5a , 0x5e , 0xda ,
    ↪ 0xde ,
0x95 , 0xd0 , 0x50 , 0x05 , 0x55 , 0x00 , 0x90 , 0xd5 ,
    ↪ 0x92 , 0x97 , 0x02 , 0x07 , 0x52 , 0x57 , 0xd2 ,
    ↪ 0xd7 ,
0x9d , 0xd9 , 0x59 , 0x0d , 0x5d , 0x09 , 0x99 , 0xdd ,
    ↪ 0x9b , 0x9f , 0x0b , 0x0f , 0x5b , 0x5f , 0xdb ,
    ↪ 0xdf ,
0x16 , 0x13 , 0x83 , 0x86 , 0x46 , 0x43 , 0xc3 , 0xc6 ,
    ↪ 0x41 , 0x14 , 0xc1 , 0x84 , 0x11 , 0x44 , 0x81 ,
    ↪ 0xc4 ,
0x1c , 0x48 , 0xc8 , 0x8c , 0x4c , 0x18 , 0x88 , 0xcc ,
    ↪ 0x1a , 0x1e , 0x8a , 0x8e , 0x4a , 0x4e , 0xca ,
    ↪ 0xce ,
0x35 , 0x60 , 0xe0 , 0xa5 , 0x65 , 0x30 , 0xa0 , 0xe5 ,
    ↪ 0x32 , 0x37 , 0xa2 , 0xa7 , 0x62 , 0x67 , 0xe2 ,
    ↪ 0xe7 ,
0x3d , 0x69 , 0xe9 , 0xad , 0x6d , 0x39 , 0xa9 , 0xed ,
    ↪ 0x3b , 0x3f , 0xab , 0xaf , 0x6b , 0x6f , 0xeb ,
    ↪ 0xef ,
0x26 , 0x23 , 0xb3 , 0xb6 , 0x76 , 0x73 , 0xf3 , 0xf6 ,
    ↪ 0x71 , 0x24 , 0xf1 , 0xb4 , 0x21 , 0x74 , 0xb1 ,
    ↪ 0xf4 ,
0x2c , 0x78 , 0xf8 , 0xbc , 0x7c , 0x28 , 0xb8 , 0xfc ,
    ↪ 0x2a , 0x2e , 0xba , 0xbe , 0x7a , 0x7e , 0xfa ,
    ↪ 0xfe ,
0x25 , 0x70 , 0xf0 , 0xb5 , 0x75 , 0x20 , 0xb0 , 0xf5 ,
    ↪ 0x22 , 0x27 , 0xb2 , 0xb7 , 0x72 , 0x77 , 0xf2 ,
    ↪ 0xf7 ,
0x2d , 0x79 , 0xf9 , 0xbd , 0x7d , 0x29 , 0xb9 , 0xfd ,
```

```
          ↪  0x2b , 0x2f , 0xbb , 0xbf , 0x7b , 0x7f , 0xfb ,
          ↪  0xff};


ROUND_CONST_BYTE RC[40] = {
       0x01, 0x03, 0x07, 0x0F, 0x1F, 0x3E, 0x3D, 0x3B, 0x37, 0
          ↪ x2F,
       0x1E, 0x3C, 0x39, 0x33, 0x27, 0x0E, 0x1D, 0x3A, 0x35, 0
          ↪ x2B,
       0x16, 0x2C, 0x18, 0x30, 0x21, 0x02, 0x05, 0x0B, 0x17, 0
          ↪ x2E,
       0x1C, 0x38, 0x31, 0x23, 0x06, 0x0D, 0x1B, 0x36, 0x2D, 0
          ↪ x1A,};
```

7.2. FELICS benchmarking results

7.3. Size on disk

Code size

SKINNY128-128

Component	ROM	text	data	bss	dec
cipher	2382	2110	272	4	2386
common	2010	2010	0	0	2010
constants	552	40	512	0	552
decryption_key_schedule	46	46	0	0	46
decrypt	46	46	0	0	46
encryption_key_schedule	46	46	0	0	46
encrypt	46	46	0	0	46
main	262	262	0	0	262
test_vectors	48	48	0	0	48

AES-128-128

Component	ROM	text	data	bss	dec
cipher	7490	7218	272	4	7494
common	2010	2010	0	0	2010
decryption_key_schedule	46	46	0	0	46
decrypt	2193	2193	0	0	2193
encryption_key_schedule	306	306	0	0	306
encrypt	2198	2198	0	0	2198
gmul_o	325	325	0	0	325
inv_sbox	256	256	0	0	256
main	262	262	0	0	262
rc_tab	10	10	0	0	10
sbox	256	256	0	0	256
test_vectors	48	48	0	0	48

Speck-64-94

Component	ROM	text	data	bss	dec
cipher	2566	2294	272	4	2570
common	593	593	0	0	593
decryption_key_schedule	46	46	0	0	46
decrypt	114	114	0	0	114
encryption_key_schedule	154	154	0	0	154
encrypt	113	113	0	0	113
main	262	262	0	0	262
test_vectors	28	28	0	0	28

Simon-64-96

Component	ROM	text	data	bss	dec
cipher	2976	2704	272	4	2980
common	593	593	0	0	593
constants	42	42	0	0	42
decryption_key_schedule	46	46	0	0	46
decrypt	192	192	0	0	192
encryption_key_schedule	324	324	0	0	324
encrypt	192	192	0	0	192
main	262	262	0	0	262
test_vectors	28	28	0	0	28

Figure 4: size on disk

7.4. Code execution time

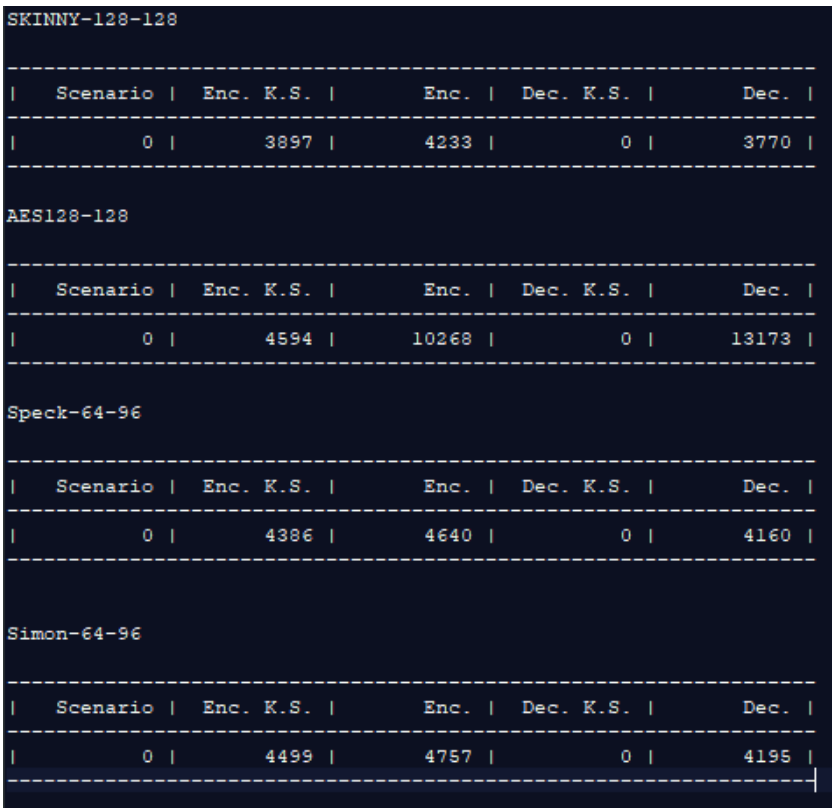

Figure 5: Code execution

SKINNY proved to be faster than AES, SPECK, and SIMON. SKINNY was created with efficiency in mind, so this result is expected.

7.5. RAM

Skinny-128-128

Block Size	Key Size	R.K.s Size	Data RAM	Scenario	Enc. K.S.	Enc.	Dec. K.S.	Dec.
16	16	320	352	4	0	0	0	0

AES-128-128

Block Size	Key Size	R.K.s Size	Data RAM	Scenario	Enc. K.S.	Enc.	Dec. K.S.	Dec.
16	16	176	208	160	17	124	0	156

Speck-64-96

Block Size	Key Size	R.K.s Size	Data RAM	Scenario	Enc. K.S.	Enc.	Dec. K.S.	Dec.
8	12	104	124	20	16	8	0	8

Simon-64-96

Block Size	Key Size	R.K.s Size	Data RAM	Scenario	Enc. K.S.	Enc.	Dec. K.S.	Dec.
8	12	168	188	24	20	17	0	17

Figure 6: RAM allocation

When compared to AES, SPECK, and SIMON, SKINNY proved to be the most expensive algorithm for use of RAM. Skinny utilises a 16 bit block size compared to Speck and Simon and this likely accounts for the increased use of RAM.

8. Evaluation

This demonstrated a benchmark of testing on a clean build for each of the encryption algorithms tested and it was found that SKINNY outperformed AES, SPECK, and SIMON in performance based testing using FELICS framework.

9. Future research

In [5] a testing solution on physical equipment is proposed. In future, testing methods used in [5] could be modified to this research to investigate hardware based encryption methods. This challenge in this case would be to accurately bench-mark data and create repeatable results.

10. Conclusion

This paper includes a literature review of lightweight encryption, and growing security concerns as our use of internet connected devices continues to grow. Proposed legislation mandating minimum security standards is attempting to close the gaps on these threats, but hostile actors continue to challenge existing protections through a variety of attacks designed to compromise the security of the device at all levels of IoT architecture [3].

11. Bibliography

[1]"What is internet of things (IoT)? - Definition from WhatIs.com", IoT Agenda, 2019. [Online]. Available: https://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT. [Accessed: 12- May- 2019].

[2]A. Baker, A SURVEY OF INFORMATION SECURITY IMPLEMENTATIONS FOR THE INTERNET OF THINGS. 2019, pp. 3 - 17.

[3]W. Buchanan, "CSN11117/CSN11102: Future Crypto", Asecuritysite.com, 2019. [Online]. Available: https://asecuritysite.com/csn11117/unit09. [Accessed: 13- May- 2019].

[4]M. Katagi and S. Moriai, Lightweight Cryptography for the Internet of Things. Sony Corporation, 2019, pp. 1 - 5

[5]Y. Kim and G. Kim, A Performance Analysis of Lightweight Cryptography Algorithm for Data Privacy in IoT Devices. DaeJon: 2018 International Conference on Information and Communication Technology Convergence (ICTC), 2018, pp. 936-938.

[6]E. Raja Naru, D. Saini and M. Sharma, A Recent Review on Lightweight Cryptography in IoT. Solan: (IoT in Social, Mobile, Analytics and Cloud, 2019, pp. 887 - 890.

[7]Secure by Design: Improving the cyber security of consumer Internet of Things Report. DCMS, 2019.

[8]"Margot James speech at the IET Conference", GOV.UK, 2019. [Online]. Available: https://www.gov.uk/government/speeches/margot-james-speech-at-the-iet-conference. [Accessed: 12- May- 2019].

[9] I. 29192-1:2012, "ISO/IEC 29192-1:2012", ISO, 2019. [Online]. Available: https://www.iso.org/standard/56425.html. [Accessed: 12- May- 2019].

[10]I. 29192-2:2012, "ISO/IEC 29192-2:2012", ISO, 2019. [Online]. Available: https://www.iso.org/standard/56552.html. [Accessed: 12- May- 2019].

[11]Nist.gov, 2019. [Online]. Available: https://www.nist.gov/sites/default/files/documents/2016/10/17/chen-presentation- lwc2016.pdf. [Accessed: 12- May- 2019].

[12]I. Andrea, C. Chrysostomou and G. Hadjichristofi, Internet of Things: Security Vulnerabilities and Challenges. 2019.

[13]A. Biryukov and L. Perrin, "State of the Art in Lightweight Symmetric Cryptography", Eprint.iacr.org, 2019. [Online]. Available: https://eprint.iacr.org/2017/511.pdf. [Accessed: 12- May- 2019].

[14]"CryptoLUX > FELICS", Cryptolux.org, 2019. [Online]. Available: https://www.cryptolux.org/index.php/FELICS. [Accessed: 12- May- 2019].

[15]"CryptoLUX > FELICS Block Ciphers Brief Results", Cryptolux.org, 2019. [Online]. Available: https://www.cryptolux.org/index.php/FELICSc Block C ciphers Brief Results. [Accessed: 13- May- 2019].

[16]"Calls for "Lightweight" Encryption are Short-Sighted and Dangerous", Private Internet Access Blog, 2019. [Online]. Available: https://www.privateinternetacc for -lightweight-encryption-are-short-sighted-and-dangerous/. [Accessed: 13- May- 2019].

[17]"Hashing", Asecuritysite.com, 2019. [Online]. Available: https://asecuritysite.com/encryp [Accessed: 13- May- 2019].

[18]C. Beierle et al., The SKINNY Family of Block Ciphers and its Low-Latency Variant MANTIS (Full Version). Nangyang, 2019, pp. 1 - 49.

[19] Jean, J., Nikolic, I., Peyrin, T.: Tweaks and keys for block ciphers: The TWEAKEY framework.In Sarkar, P., Iwata, T., eds.: ASIACRYPT 2014, Part II. Volume 8874 of LNCS., Springer, Heidelberg (December 2014) 274–288

.

References

[1] Bresson, E., Chevassut, O., and Pointcheval, D. Dynamic group diffie-hellman key exchange under standard assumptions. In *International conference on the theory and applications of cryptographic techniques* (2002), Springer, pp. 321–336.

[2] Megahed, M. H., Makrakis, D., and Ying, B. Survsec: A new security architecture for reliable network recovery from base station failure of surveillance wsn. *Procedia Computer Science 5* (2011), 141–148. The 2nd International Conference on Ambient Systems, Networks and Technologies (ANT-2011) / The 8th International Conference on Mobile Web Information Systems (MobiWIS 2011).

[3] Rani, D. J., and Roslin, S. E. Light weight cryptographic algorithms for medical internet of things (iot)-a review. In *2016 Online International Conference on Green Engineering and Technologies (IC-GET)* (2016), IEEE, pp. 1–6.