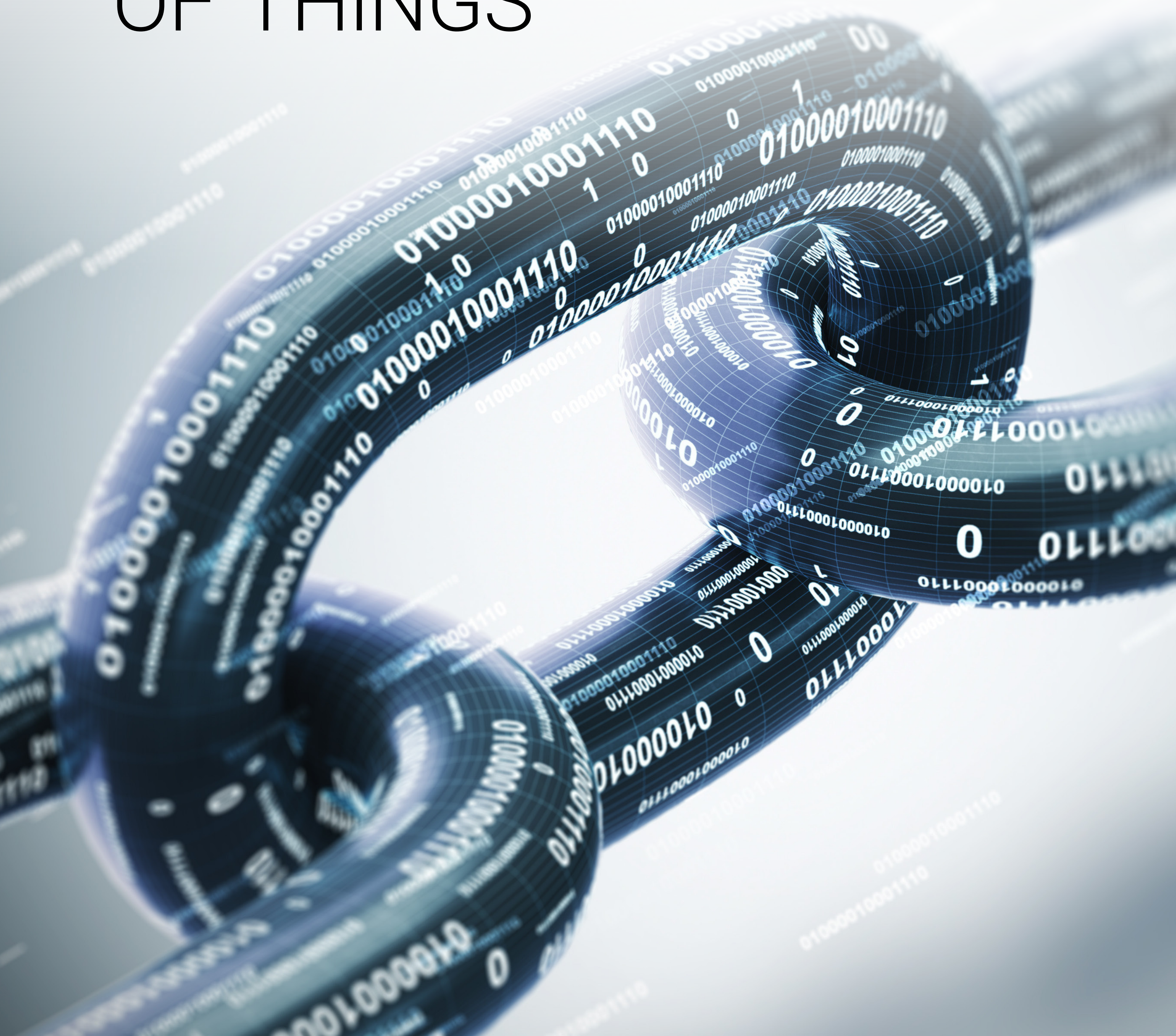
CyberusLabs

SECURITY • ENCRYPTION • PERFORMANCE

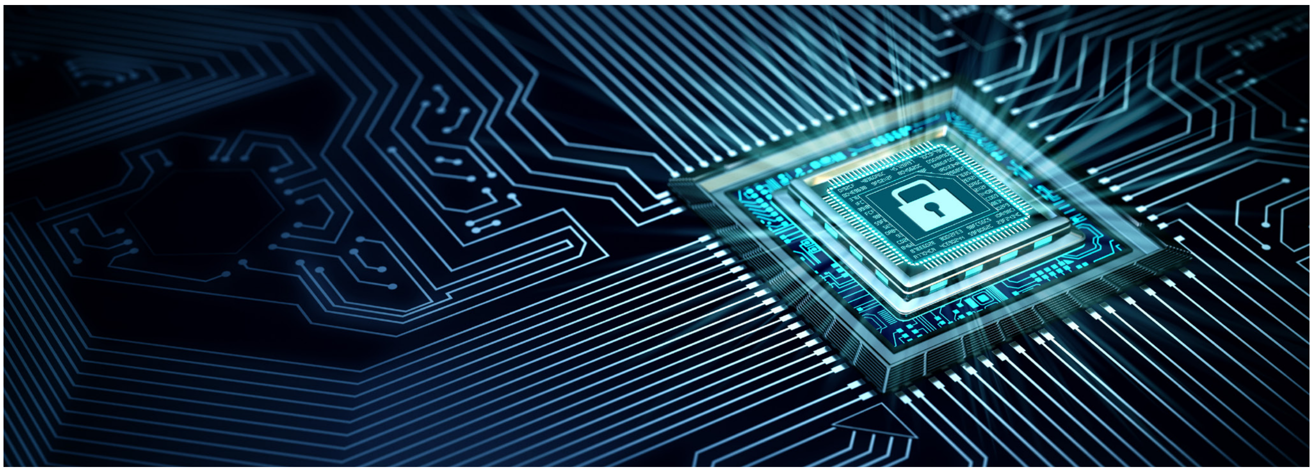
Cybpher – A lightweight Binary Substitution Cipher

ENCRYPTING THE INTERNET OF THINGS



INTRODUCTION

The Lightweight Encryption (LE) algorithm, internally referred to a Cybpher is a part of the Horizon 2020 funded Cyberus Labs ELIoT Pro project and comprises a critical part of the overall ELIoT Pro IoT Cyber Security Platform. ELIoT Pro contains a user authentication component designed to secure the Human to Machine (H2M) communications, a Machine to Machine (M2M) component to secure all M2M communications and an IoT Self Healing component.



The M2M communications and device authentication component is designed to ensure that IoT data is fully encrypted and sent only to authorized devices. LE is the M2M secure communications and device authentication component of the ELIoT Pro Platform.

LE is the ELIoT Pro's low compute and low power resource solution to the limitations of today's IoT environments. Current and future IoT applications use small battery powered sensors and actuators, so encryption and M2M authentication must operate on extremely low power and computational resources. The small processor size limits and low clock speeds typically encountered in IoT implementations combined with limited battery power have greatly

limited encryption and authentication options available to the IoT industry. PKI computational requirements have all but eliminated its use in IoT and have created a dearth of available cyber security solutions for the industry. This has resulted in a need for lightweight, high security encryption and authentication protocols. In Oct. 2016 NIST had commissioned a call for papers on Lightweight Encryption and a number of submissions were presented.
LE development was informed in part by the requirements and results of the NIST study.

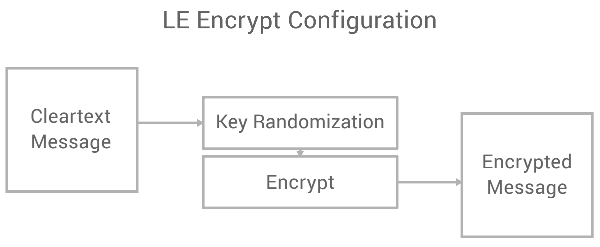
ELIoT Pro's LE was formulated, built and tested to satisfy the key requirements of being a computationally light, low memory use cypher which was at once simple and highly secure. The binary substitution Vernam based approach to LE does not rely on polynomial computing normally used for prime number operations in modern encryption algorithms. This has the advantage of fast, low CPU usage operation, while eliminating LE vulnerability to computationally intensive brute force attacks. These types of attacks render conventional encryption vulnerable, but they are not effective against LE because LE does not encrypt using mathematical operations. In the coming age of Quantum Computing, compute intensive brute force attacks are of great concern for the cryptographic community and cyphers resistant to these attacks are in high demand.

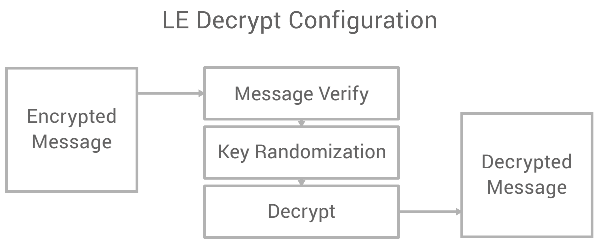
The LE key generation algorithm, operating on both ends of an LE transmission link, produces a two peer blockchain implementation wherein both peers contain a type of one way hash of all preceding communications between a pair of devices. This creates a pair of entangled peer transmission points. The result is a hyper secure communications flow between peers, which continuously authenticates both sides of the transmission via continuing, growing entanglement of the peers. In this manner LE is able to secure IoT transmissions while it authenticates IoT devices on its network.

CORE ALGORITHM

The Cybpher algorithm is a lightweight binary substitution cipher, more specifically, it is an iterated polyalphabetic substitution block encryption variant, employing principles of a modified Vernam cipher. The algorithm can encrypt and decrypt both: data streams as well as data blocks.

Encrypted content exhibits consistent encryption entropy and encryption/decryption performance that is independent of the type of input data (See Table 1-3). There are no input data size restrictions (No required block size) which avoids potential block padding issues that exist with traditional block ciphers.





When encrypting/decrypting the bytes of the message are processed sequentially with minimal computational overhead since only addition/subtraction instructions and memory/register access are needed. The minimum processor requirements are 8-bit, the minimum memory requirements are:
(Processor bit width * 256 * 3) + 4 bytes for counters

A 4-bit variant with much lower memory and compute requirements is currently undergoing testing. It retains the original strong encryption characteristics with somewhat lower entropy characteristics. If additional redundancy and improved recovery from transmission errors is required, the above memory requirement can be doubled to take advantage of the built-in recovery mechanisms.

Where a per-message MAC is required, Cybpher can calculate a MAC value with little computational overhead and return it as part of the encryption/decryption functions. The MAC is 16-bits and satisfies both data integrity and message authenticity.

PERFORMANCE TESTING

To evaluate the performance of the Cybpher algorithm, we compared our encryption/decryption times and overall encryption entropy against the industry standard AES 128bit and AES 256bit encryption algorithm.

- AES 128bit is identical to Rijndael 128bit
- AES 256bit is a modified version of the Rijndael 256bit algorithm. AES 256 differs from the original Rijndael in that the AES implementation uses a key size of 256 bit but only a block size of 128 bit.

The tests were conducted on two platforms, an embedded system with a 32-bit ARM Cortex A8 processor running Debian 7.11 and a 64-bit Intel i7 desktop system running Ubuntu 18.04.

Embedded system specifications:

CPU	AM335x 1GHz ARM® Cortex-A8 https://www.ti.com/product/am3358
RAM	512MB DDR3
Storage	4GB eMMC
FPU	NEON floating-point accelerator
OS	Debian GNU/Linux 7.11

Desktop system specifications:

CPU	Intel Core i7, 8700k @ 3.7 GHz https://en.wikichip.org/wiki/intel/core_i7/i7-8700k
RAM	16GB DDR4
Storage	1TB SSD
FPU	Integrated x87 FPU
OS	Ubuntu 18.04.01

The tests included several different types of input data, different input sizes and different processing iterations ranging from 1 to 1000. We measured combined encrypt/decrypt times as well as separate profiling for encryption and decryption. NOTE: Unless stated otherwise, throughput values are in Mbps (Megabits (one million) bits per second).

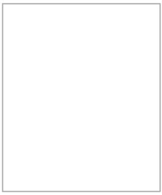
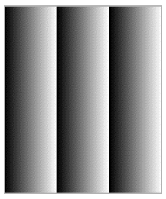
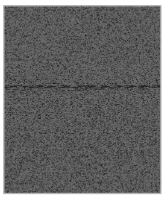
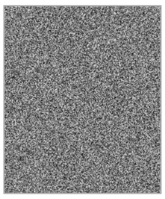
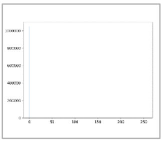
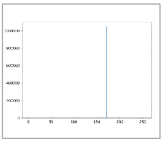
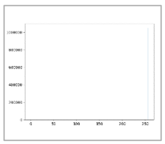
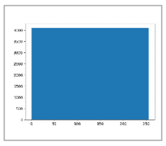
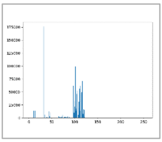
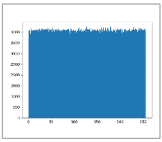
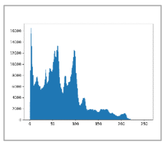
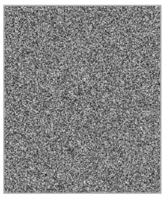
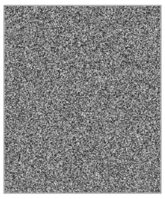
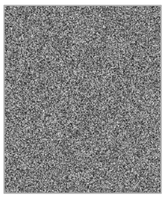
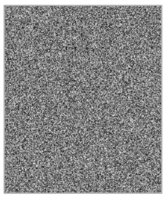
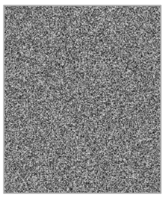
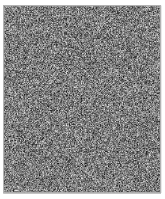
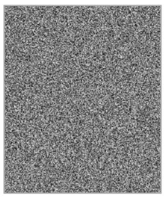
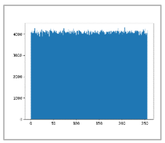
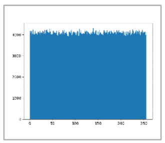
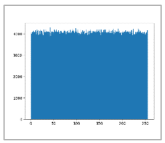
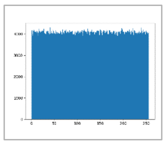
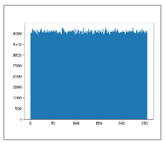
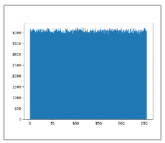
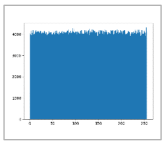
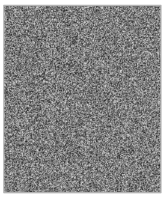
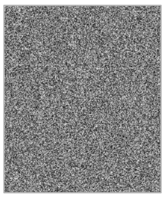
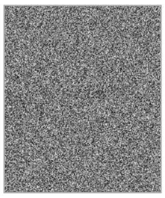
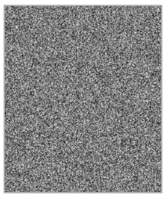
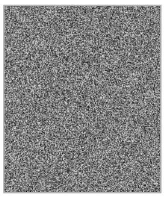
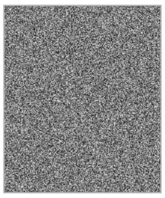
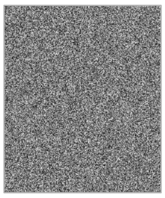
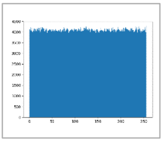
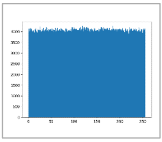
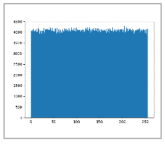
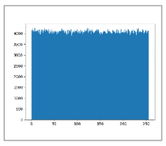
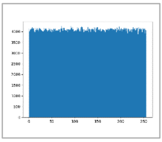
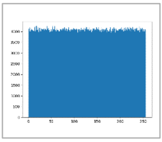
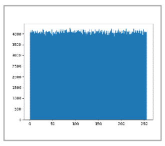
Table 1: Embedded ARM system profiling

Algorithm	Compiler options	Input size (bytes)	Iterations	Avg. encryption time	Avg. decryption time	Avg. time total	Throughput Mbps
Cybpher	-DNDEBUG -O3	1,048,576 (1 MB)	1	54 ms	39 ms	93 ms	86.02
AES 128	-DNDEBUG -O3	1,048,576 (1 MB)	1	128 ms	85 ms	213 ms	37.56
AES 256	-DNDEBUG -O3	1,048,576 (1 MB)	1	139 ms	100 ms	239 ms	33.47
Cybpher	-DNDEBUG -O3	1,048,576 (1 MB)	1000	25,506 ms	20,217 ms	45,723 ms	174.97
AES 128	-DNDEBUG -O3	1,048,576 (1 MB)	1000	81,082 ms	86,198 ms	167,280 ms	47.82
AES 256	-DNDEBUG -O3	1,048,576 (1 MB)	1000	95,910 ms	100,732 ms	196,642 ms	40.68

Table 2: Desktop Intel i7 system profiling

Algorithm	Compiler options	Input size (bytes)	Iterations	Avg. encryption time	Avg. decryption time	Avg. time total	Throughput Mbps
Cybpher	-DNDEBUG -O3	1,048,576 (1 MB)	1	2.2 ms	1.6 ms	3.8 ms	2,105.26
AES 128	-DNDEBUG -O3	1,048,576 (1 MB)	1	5.2 ms	5.2 ms	10.4 ms	769.23
AES 256	-DNDEBUG -O3	1,048,576 (1 MB)	1	6.5 ms	6.5 ms	13.0 ms	615.38
Cybpher	-DNDEBUG -O3	1,048,576 (1 MB)	1000	2,221 ms	1,569 ms	3,790 ms	2,110.82
AES 128	-DNDEBUG -O3	1,048,576 (1 MB)	1000	5,254 ms	5,207 ms	10,461 ms	764.75
AES 256	-DNDEBUG -O3	1,048,576 (1 MB)	1000	6,504 ms	6,540 ms	13,044 ms	613.31

Table 3: Encryption entropy visualization

	Input buffer filled with all 0 (0x00)	Input buffer filled with all 170 (0xAA)	Input buffer filled with all 255 (0xFF)	Input buffer filled with repeating 0-255 values	Input buffer filled with ASCII text	Input buffer filled with random bytes	Input buffer filled with image data
Unmodified input buffer visualization							
Unmodified input buffer histogram							
Cybpher encrypted buffer visualization							
Cybpher encrypted buffer histogram							
AES 256bit encrypted buffer visualization							
AES 256bit encrypted buffer histogram							

CyberusLabs

Bringing cybersecurity to the next level

Contact us

Marek Ostafil, COO
+48 692 437 857
marek.ostafil@cyberuslabs.com

Cyberus Labs Sp z o.o.
ul. Warszawska 6/309
40-006 Katowice, Poland
www.cyberuslabs.com

The project ELIoT Pro has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No 822641