Algoritmos Leves de Criptografia

Henrique Macedo

henriquebmacedo@gmail.com

Agenda

- Triathlon of Lightweight Block Ciphers for the Internet of Things
- Projeto BLOC
- Comparação Projeto BLOC x Triathlon
- Descrição de ambientes de desenvolvimento
- Estágio atual do trabalho

Agenda

- **Triathlon of Lightweight Block Ciphers for the Internet of Things**
- Projeto BLOC
- Comparação Projeto BLOC x Triathlon
- Descrição de ambientes de desenvolvimento
- Estágio atual do trabalho

Artigo

- **Nome**

  - Triathlon of Lightweight Block Ciphers for the Internet of Things

- **Autores**

  - Daniel Dinu, Yann Le Corre, Dmitry Khovratovich, Leo Perrin, Johann Groschadl, Alex Biryukov

Triathlon: Implementação

- **Foram implementados 13 algoritmos**

  - AES, Fantomas, HIGHT, LBlock, LED, Piccolo, PRESENT, PRINCE, RC5, Robin, Simon, Speck, and TWINE.

- **Linguagem escolhida:** C
- **Foram usadas 3 plataformas diferentes**

  - 8-bit AVR ATmega128
  - 16-bit TI MSP430F1611
  - 32-bit ARM Cortex-M3

Triathlon: Métricas

- **Foram consideradas 3 métricas**

  - Tamanho do código binário

  Medido em bytes, corresponde ao programa armazenado na memória Flash do dispositivo alvo.

  - Consumo de memória RAM

  Dividido em consumo de dados e consumo do *stack*.

  - Tempo de execução

  Número de ciclos do processador gastos executando um conjunto de instruções.

Triathlon: Algoritmos

Cipher	Year	Block Size	Key Size	Round Keys Size	Rounds
AES	1998	128	128	1408	10
Fantomas	2014	128	128	0	12
HIGHT	2006	64	128	1088	32
Lblock	2011	64	80	1024	32
LED	2011	64	80	0	48
Piccolo	2011	64	80	864	25
PRESENT	2007	64	80	2048	31
PRINCE	2012	64	128	192	12
RC5	1994	64	128	1344	20
Robin	2014	128	128	0	16
Simon	2013	64	96	1344	42
Speck	2013	64	96	832	26
TWINE	2011	64	80	1152	36

Triathlon: Cenários

- **Cenário 1 – Protocolo de Comunicação**

  - Cobre a necessidade de uma comunicação segura.

  - Seguindo um padrão do IEEE e ZigBee, o tamanho da mensagem transmitida é de 128 bytes.

  - A chave mestra é armazenada na RAM e as chaves de round são computadas e depois armazenadas na RAM.

  - Encriptação é feita com objetivo de reduzir o consumo de memória RAM.

Triathlon: Cenários

- **Cenário 2 - *Challenge-Handshake Authentication Protocol***

- Cobre a necessidade de autenticação na Internet das Coisas

- Dados de 128 bits

- Adequado para ambientes muito restritos onde tamanho de código e consumo de RAM devem ser muito baixos enquanto o tempo de execução tem que ser rápido para evitar desperdício de bateria.

Triathlon: Benchmarking

- **Benchmarking Framework**

- Trabalhos anteriores avaliam performance em diferentes plataformas usando diferentes condições de medidas.

- As conclusões não são precisas e não inspiram confiança.

- Isso tudo serviu de motivação para criarem um framework que tornasse justas as comparações.

Triathlon: Metodologia

- **Metodologia 1**

$$p_{i,d} = \sum_{m \in M} w_m \frac{v_{i,d,m}}{\min_i(v_{i,d,m})}$$

$p_{i,d}$: Parâmetro de performance

w_m : Peso relativo à métrica m

$v_{i,d,m}$: Valor da métrica m para a implantação i na plataforma d

Triathlon: Metodologia

- **Metodologia 1**

- *Figure-of-Merit* (FOM): Valor médio da performance nas 3 plataformas.

$$\mathrm{FOM}(i_1, i_2, i_3) = \frac{p_{i_1,AVR} + p_{i_2,MSP} + p_{i_3,ARM}}{3}$$

Triathlon: Metodologia 1

Cipher	AVR			MSP			ARM			FOM
	Code Size [Bytes]	RAM [Bytes]	Execution Time [cycles]	Code Size [Bytes]	RAM [Bytes]	Execution Time [cycles]	Code size [Bytes]	RAM [Bytes]	Execution Time [cycles]	
I: Encryption + Decryption (including key schedule)										
Speck	**1692**	300	239532	**1342**	300	93239	**792**	332	**19461**	3.8
Robin	4950	266	146173	3170	238	76878	3668	**304**	92150	6.8
Fantomas	5898	262	111701	4164	**234**	57430	4604	308	70042	7.1
Simon	2476	375	390119	8158	392	214745	892	400	25690	7.4
RC5	2616	377	515864	1952	378	482894	1144	408	32865	8.1
LBlock	3086	331	207631	2024	328	313349	2136	406	162576	8.7
HIGHT	2608	342	168569	2368	342	423221	2196	392	173589	9.5
Piccolo	2654	319	407931	1824	318	349423	1604	406	291330	11.4
PRINCE	5650	**241**	280381	4174	240	405552	4660	392	226333	12.4
TWINE	3610	402	384993	3452	352	565495	2464	418	256997	13.2
AES	26800	551	**109446**	20726	574	**54075**	15256	576	40820	20.4
LED	4538	274	2634460	7004	252	2505640	3732	334	692194	40.6
PRESENT	11208	591	3838051	12928	1042	2864032	7372	790	606922	51.4

Triathlon: Metodologia 1

Cipher	AVR			MSP			ARM			FOM
	Code Size [Bytes]	RAM [Bytes]	Execution Time [cycles]	Code Size [Bytes]	RAM [Bytes]	Execution Time [cycles]	Code Size [Bytes]	RAM [Bytes]	Execution Time [cycles]	
Balanced (globally efficient)										
Speck	**572**	**49**	14003	**618**	58	6054	**512**	**96**	**904**	3.9
Simon	878	65	24305	940	82	12902	600	104	1376	6.1
RC5	804	59	22395	700	**54**	20543	628	104	1730	6.7
Fantomas	2400	103	**5866**	1920	78	3646	2184	184	4552	8.1
Robin	2434	103	7760	1942	80	4935	2188	184	6187	9.0
LBlock	1320	59	11119	976	58	18988	1192	148	10215	10.0
HIGHT	1084	54	11399	980	62	26728	1008	128	11602	11.0
Piccolo	1178	65	25681	966	70	21448	940	160	18388	14.0
TWINE	1408	59	21637	1570	72	34778	1180	156	15677	14.9
AES	2742	127	22603	8844	92	**2862**	7360	184	2418	15.7
PRINCE	4300	63	17207	3418	70	25340	4076	224	14344	17.9
PRESENT	5172	193	203237	4960	396	115338	3520	260	26279	47.3
LED	2482	86	143253	4422	104	148334	2212	192	41728	48.0

Triathlon: Metodologia

- **Metodologia 2**

$$p_{i,d} = \sum_{m \in \{\text{code, RAM}\}} w_m \frac{v_{i,d,m}}{\max_i(v_{i,d,m})}$$

$p_{i,d}$: Parâmetro de performance

w_m : Peso relativo à métrica m

$v_{i,d,m}$: Valor da métrica m para a implantação i na plataforma d

Triathlon: Metodologia 2

Cipher	AVR			MSP			ARM		
	Code Size [Bytes]	RAM [Bytes]	Execution Time [cycles]	Code Size [Bytes]	RAM [Bytes]	Execution Time [cycles]	Code Size [Bytes]	RAM [Bytes]	Execution Time [cycles]
I: Small code size & RAM									
AES	2742	127	22603	3124	124	33386	2444	232	19735
Fantomas	2400	103	**5866**	1920	78	**3646**	2184	184	4552
HIGHT	1084	54	11399	980	62	26728	1008	128	11602
LBlock	1268	**46**	16473	976	58	18988	1192	148	10215
LED	2482	86	143253	4042	96	694812	2212	192	41728
PRESENT	1562	83	1937461	3650	352	3497578	1760	280	221471
PRINCE	4300	63	17207	3418	70	25340	4076	224	14344
Piccolo	1178	65	25681	966	70	21448	940	160	18388
RC5	804	59	22395	700	**54**	20543	628	104	1730
Robin	2434	103	7760	1942	80	4935	2188	184	6187
Simon	878	65	24305	940	82	12902	600	104	1376
Speck	**572**	49	14003	**618**	58	6054	**512**	**96**	**904**
TWINE	1408	59	21637	1570	72	34778	1180	140	20505

Triathlon: Conclusão

- O ranking geral do cenário 1 é semelhante ao do cenário 2.
- O Top-7 é o mesmo para os 2: Speck, Robin, Fantomas, Simon, RC5, LBlock and Hight.
- Houve caso de uma mesma implementação ter resultado geral melhor na plataforma de 8-bits do que na de 16-bits.
- AES é o algoritmo mais rápido em geral.

Agenda

- Triathlon of Lightweight Block Ciphers for the Internet of Things
- **Projeto BLOC**
- Comparação Projeto BLOC x Triathlon
- Descrição de ambientes de desenvolvimento
- Estágio atual do trabalho

Projeto BLOC: Algoritmos

- **Foram implementados 21 algoritmos**

- *Classic Block Ciphers*: AES, NOEKEON, IDEA, SKIPJACK

- *Lightweight Block Ciphers*: DESXL, HIGHT, KATAN and KTANTAN, KLEIN, LBlock, LED, mCrypton, MIBS, Piccolo, PRESENT, SEA, TEA and XTEA, TWINE, SIMON and SPECK.

Projeto BLOC: Implementação

- **Os códigos em C foram desenvolvidos por 4 pessoas:**

  Mickaël Cazorla, Sylvain Gourgeon, Kevin Marquet and Marine Minier.

- Foram implementados para o sensor **WSN430** que é baseado no microcontrolador de 16 bits **MSP430 (mesmo microcontrolador de 16 bits do Triathlon)**.
- Todos os códigos estão disponibilizados num repositório GitHub.

Projeto BLOC: Otimização

- **Os códigos foram compilados com 5 opções:**

  - Utilizando o parâmetro –Os
  - Utilizando o parâmetro –O0
  - Utilizando o parâmetro –O1
  - Utilizando o parâmetro –O2
  - Utilizando o parâmetro –O3

- Para cada parâmetro foi gerada uma tabela com muitos dados contendo tempo de execução e memória usada por cada algoritmo.

Agenda

- Triathlon of Lightweight Block Ciphers for the Internet of Things
- Projeto BLOC
- **Comparação Projeto BLOC x Triathlon**
- Descrição de ambientes de desenvolvimento
- Estágio atual do trabalho

Projeto BLOC x Triathlon

- LED – 64 bits

	BLOC	Triathlon
Top-3 Tempo de execução (ciclos)	1 – 132.136 2 – 416.546 3 – 419.408	1 – 148.334 2 – 694.812 3 – 1.186.231

Projeto BLOC x Triathlon

- PRESENT – 64 bits

	BLOC	Triathlon
Top-3 Tempo de execução (ciclos)	1 – 172.091 2 – 175.271 3 – 194.165	1 – 115.338 2 – 922.263 3 – 3.497.578

Projeto BLOC x Artigo

- AES – 128 bits

	BLOC	Triathlon
Top-3 Tempo de execução (ciclos)	1 – 10.711 2 – 11.850 3 – 11.936	1 – 2.862 2 – 22.766 3 – 33.386

Projeto BLOC x Triathlon

- **Conclusão**

- As implementações do BLOC de: HIGHT, LBlock, Piccolo e Twine são ligeiramente piores que as do Triathlon.

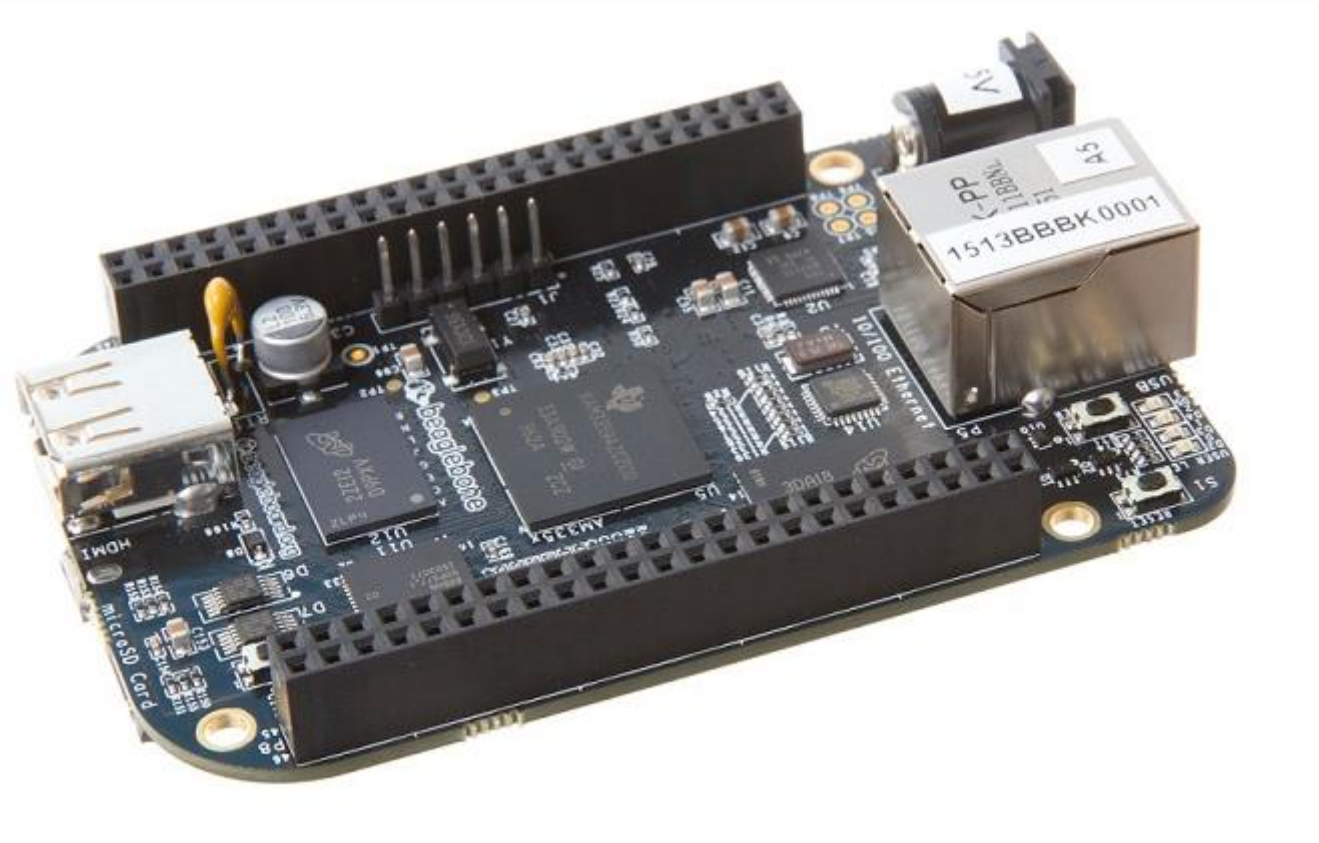
- As implementações do BLOC de AES e PRESENT são bem mais rápidas do que as do Triathlon.

- Framework mais restrito seguido pelo Triathlon pode ter tido influência nesses resultados.

Agenda

- Triathlon of Lightweight Block Ciphers for the Internet of Things
- Projeto BLOC
- Comparação Projeto BLOC x Triathlon
- **Descrição de ambientes de desenvolvimento**
- Estágio atual do trabalho

Beaglebone Black



Beaglebone Black

Processador	**Sitara ARM Cortex–A8, 32 bits 1GHz, 2000 MIPS**
Engine Gráfica	SGX530 3D, 20M Polígonos/s
Memória SDRAM	512MB DDR3L 800MHz
Onboard Flash	4GB
Outras features	microHDMI, microSD, USB, Ethernet

BeagleBone Black - PRUSS

- **Progammable Realtime Unit SubSystem (PRUSS):**

  - 2 PRUs (PRU0 e PRU1) com suas respectivas memórias associadas.

  - As 2 PRUs podem trabalhar completamente independentes.

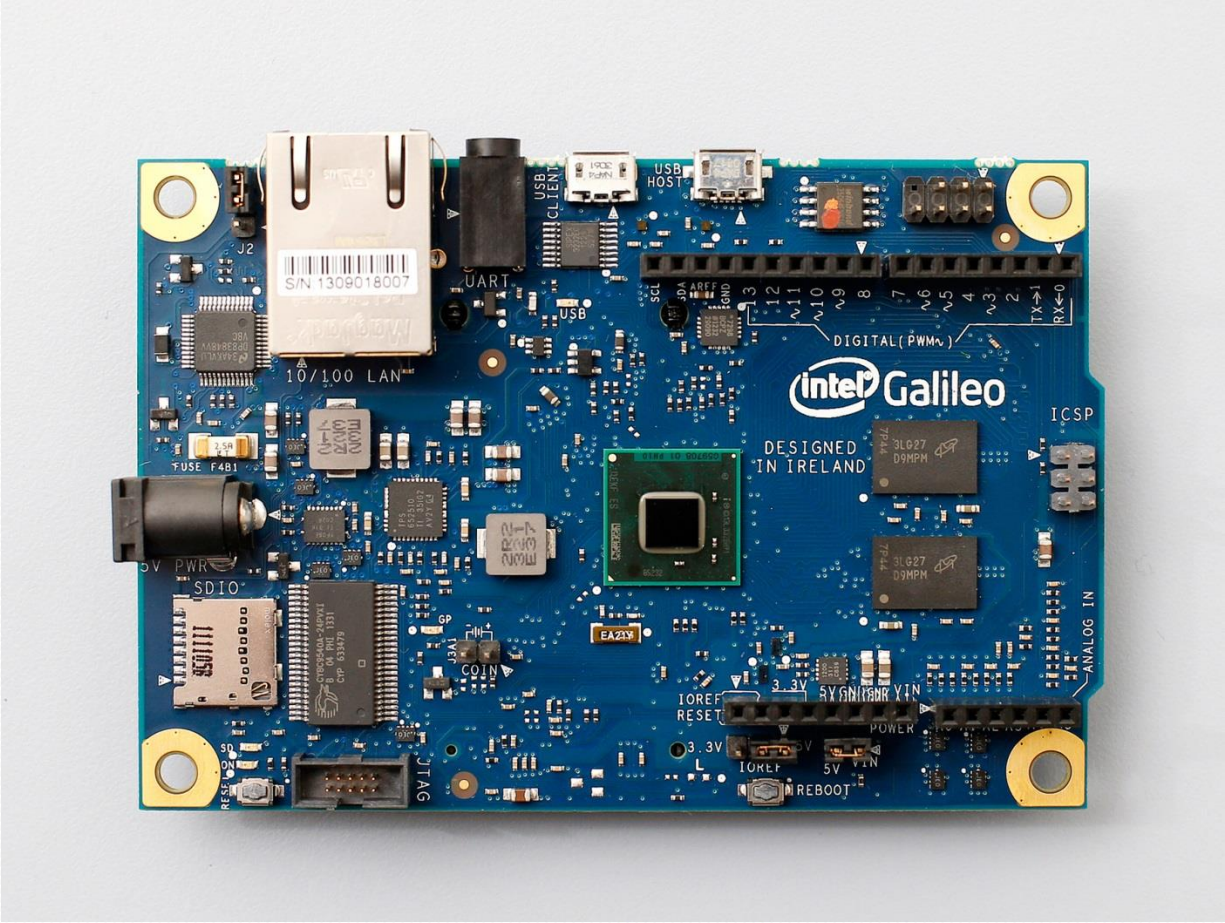
  - Eficiente na manipulação de eventos do sistema que têm limitações apertadas de tempo real.

BeagleBone Black - PRUSS

- **Características**

  - 32-bit Load/Store arquitetura RISC
  - 4K Byte instruction RAM por núcleo
  - 512 Bytes data RAM por núcleo
  - PRUSS pode ser desabilitada via software
  - 200 MHz clock, cada instrução toma 1 ciclo (5ns)

Galileo


S/N 1309018007
10/100 LAN
UART
USB
USB HOST
DIGITAL(PWM~)
intel Galileo
DESIGNED IN IRELAND
ICSP
SDIO
JTAG
COIN
IOREF
RESET
3.3V
5V
REBOOT
POWER
ANALOG IN

Galileo

Processador	Intel® Quark SoC X1000 , 400 MHz 32-bit
Arquitetura	Intel ® Pentium® Class
Memória Flash	8 MB
Memória RAM	512KB on-chip SRAM, 256 MB DRAM
Sistema Operacional	Arduino Linux Distribution for Galileo
Outras features	Ethernet, USB Client, USB Host, microSD

Galileo

- Ambientes de desenvolvimento

Arduino*

Eclipse* (C/C++)

Intel XDK IoT Edition (JavaScript)

Agenda

- Triathlon of Lightweight Block Ciphers for the Internet of Things
- Projeto BLOC
- Comparação Projeto BLOC x Triathlon
- Descrição de ambientes de desenvolvimento
- **Estágio atual do trabalho**

Estágio atual do trabalho

- Com a análise dos trabalhos encontrados foi possível esclarecer os parâmetros que são pertinentes para realizar a comparação de performance entre os algoritmos.
- Foram coletados materiais sobre a PRUSS da BeagleBone Black tais como *datasheet* e *instruction set*.
- Através de tutorias na internet consegui configurar a Beaglebone para habilitar a PRUSS e a mesma já está funcionando adequadamente.

Próximos passos

- Dominar o ambiente de programação da PRUSS da BBB e o assembly da Galileo.
- Implementar o algoritmo PRESENT em ambas as plataformas buscando uma implementação menor e/ou mais rápida que as existentes.
- Fazer uma comparação da performance deste algoritmo entre os dois ambientes de desenvolvimento.

Obrigado