Comparativo de Desempenho e Consumo Energético de Algoritmos de Hash e Criptografia em Processadores ARM

Maykon Valério da Silva, Marcio Seiji Oyamada e Luiz Antonio Rodrigues
Programa de Pós-Graduação em Ciência da Computação - PPGComp
Universidade Estadual do Oeste do Paraná – UNIOESTE
Cascavel - PR - Brasil
{first.last}@unioeste.br

***Resumo*—Com a expansão da Internet das Coisas, bilhões de novos dispositivos estarão conectados em rede nos próximos anos, coletando e transmitindo dados que exigem mecanismos seguros de comunicação. No entanto, estes equipamentos possuem capacidade de processamento e memória limitados, além de restrições quanto ao consumo de energia de baterias. Este artigo compara o desempenho de execução e o consumo energético dos algoritmos de *hash* SHA-2 e SHA-3 de 256 bits, e dos algoritmos de criptografia RSA e ECC em dispositivos embarcados com processadores da família ARM (Advanced RISC Machine) big.LITTLE, que possuem *cores* de alta performance e *cores* mais simplificados. Os testes foram realizados com a plataforma de simulação Gemstone. As implementações escolhidas dos algoritmos a serem testados foram as disponibilizadas pela biblioteca wolfSSL. Os resultados mostram que os algoritmos de *hash* possuem desempenho de tempo de execução semelhante em ambos os *cores*, mas com eficiência energética superior nos *cores* LITTLE, consumindo em média 90% menos energia. Já os algoritmos de assinatura possuem desempenho superior nos *cores* big, porém o consumo energético é melhor nos *cores* LITTLE, consumindo aproximadamente 50% menos energia.**

***Index Terms*—IoT, Internet of Things, Gem5, Benchmark, Odroid-XU3.**

I. Introdução

O número de dispositivos para Internet das Coisas (IoT) ativos deve ultrapassar os 10 bilhões em 2021, devendo chegar a 25 bilhões em 2030 [1], com uso expressivo em sistemas veiculares, automação residencial, equipamentos industriais e de saúde (*Healthcare*), incluindo sistemas cíber-físicos e cíber-humanos. Com o crescimento do uso de redes IoT, cresce também a preocupação com a segurança das aplicações.

Para prover a interoperabilidade de tamanha quantidade de dispositivos heterogêneos, assim como nas redes de computadores, uma arquitetura em camadas torna o sistema mais flexível e facilita o desenvolvimento de novas soluções. Embora não se tenha um modelo de referência como o ISO/OSI, o modelo de cinco camadas é um dos mais citados na literatura, a exemplo das arquiteturas definidas por [2] e [3], que incluem, de baixo para cima, as camadas de Percepção, Rede, Middleware, Aplicação e Negócios.

Os dispositivos da camada de Percepção, onde estão conectados os sensores e atuadores, são sistemas embarcados com recursos limitados, isto é, com baixo poder de processamento, baixa quantidade de memória e, muitas vezes, alimentados por baterias, necessitando de otimização de energia.

Como qualquer sistema com comunicação entre dispositivos, as redes IoT podem se utilizar de métodos criptográficos, incluindo *hash* e assinatura digital, para garantir integridade, autenticidade e confidencialidade dos dados transmitidos.

No entanto, considerando as restrições do hardware para IoT, deve-se buscar um balanceamento entre o nível de segurança que os diversos algoritmos de *hash* e criptografia trazem e o quanto desses recursos são consumidos por estes.

Para estudos e análises de diferentes arquiteturas computacionais os simuladores se destacam pela liberdade na exploração de opções sem a necessidade de se ter os recursos de *hardware* a disposição, podendo ainda investigar o impacto de alterações nestas arquiteturas, algo que é de uma dificuldade muito grande em dispositivos e plataformas reais.

Dentre as ferramentas disponíveis para simulação em sistemas embarcados, o Gem5 apresenta-se como um dos mais utilizados. Ele é a junção das melhores características dos simuladores M5 e GEMS. O primeiro é um *framework* completo para simulação de modelos de CPUs e o segundo possui um sistema detalhado para simulações de memória [4].

Embora seja possível fazer estimativas de potência e consumo energético no Gem5, é necessário um conhecimento completo das características físicas do dispositivo sendo simulado. Desta forma, neste trabalho foi utilizado o *framework* Gemstone [5], que utiliza como dados de entrada os resultados do Gem5 e aplica técnicas estatísticas e de *machine-learning* para fazer com que plataformas de hardware possam ser levadas aos modelos do Gem5, sem a dependência de especificações detalhadas.

Desta forma, este trabalho apresenta uma comparação de desempenho, com especial atenção ao consumo energético, de algoritmos clássicos de *hash* e criptografia quando executados em uma plataforma de dispositivo embarcado simulada no Gemstone. Para comparação entre algoritmos de *hash* foram escolhidos o SHA-2 e o SHA-3 [6], ambos com 256 bits de comprimento. Já para a comparação entre algoritmos de assinatura foram escolhidos o RSA (*Rivest-Shamir-Adleman*)

[7] e o ECC (*Elliptic Curve Cryptography*) [8].

O restante do artigo está organizado da seguinte ordem. A Seção II apresenta o referencial teórico, incluindo os algoritmos de *hash* e criptografia utilizados no trabalho, e seus propósitos em uma comunicação segura. Na Seção III são apresentados alguns trabalhos relacionados. A Seção IV descreve a metodologia utilizada nos experimentos. A Seção V relata os resultados obtidos e a discussão dos mesmos. Por fim, a conclusão está na Seção VI, que apresenta também sugestões de trabalhos futuros.

II. Conceitos Teóricos

Esta seção apresenta brevemente a arquitetura IoT e os algoritmos de criptografia e *hash*.

A. Arquitetura IoT

A divisão da arquitetura para IoT em camadas tem diferentes proposições na literatura [9]. A proposta de [2] e [3] divide o modelo em cinco camadas, a saber:

1) **Camada de Percepção:** Camada de dispositivos embarcados, onde estão os sensores e atuadores;
2) **Camada de Rede:** Transporta as informações obtidas pela camada de percepção para a camada de *Middleware* através de um meio de transporte como: WiFi, Bluetooth, LPWAN (*Low Power Wide Area Network*), redes móveis (como 3G, 4G ou 5G por exemplo) e protocolos como IPV4 e MQTT;
3) **Camada de *Middleware*:** Camada que processa a informação recebida pela camada de rede, consiste normalmente de computação em nuvem;
4) **Camada de Aplicação:** Compreende um sistema IoT, como uma casa inteligente, um sistema de transporte inteligente, etc.;
5) **Camada de Negócios:** Camada que gerencia as aplicações e serviços do IoT. Aqui são gerados os modelos de negócios e as pesquisas relacionados ao IoT.

No modelo de cinco camadas, a camada de Negócios atua como a camada superior, responsável pelo controle e gerenciamento de serviços IoT fornecidos pela camada de Aplicação. A camada de *Middleware* atua como uma camada intermediária, integrando as diferentes tecnologias de rede e transportando os dados para dentro de um ambiente de nuvem.

De acordo com [10], os desafios de segurança em IoT estão relacionados à tecnologia (redes sem-fio, escalabilidade, energia, ambiente distribuído) e aos conceitos tradicionais de autenticação, confidencialidade, integridade e disponibilidade.

B. Criptografia

A segurança em uma comunicação é definida por [11] de forma didática como a habilidade de saber se a mensagem recebida está completa (integridade), se ela foi enviada realmente por quem diz ter enviado (autenticidade), e que seu conteúdo seja secreto, de forma que nenhuma outra parte tenha acesso ao conteúdo a não ser que ela seja um dos destinatários da mensagem (confidencialidade).

Nas redes de computadores esses mecanismos são fornecidos pelos algoritmos de criptografia, *hash* e assinatura digital.

Os algoritmos de *hash* são responsáveis por garantir integridade. Eles recebem uma entrada m e calculam uma cadeia de tamanho fixo $H(m)$, conhecida como *hash*. Um remetente deve calcular o *hash* de sua mensagem e enviá-lo junto com ela. Ao receber a mensagem, um receptor deve recalcular o *hash* e compará-lo com o *hash* enviado pelo remetente. Se os dois forem iguais temos um certo nível de garantia de que a mensagem não foi alterada no caminho.

Desta forma, o nível de segurança fornecido por estes algoritmos é determinado por três características:

1) Não deve ser possível calcular a entrada m a partir de $H(m)$;
2) Uma mínima mudança na entrada m deve resultar em um $H(m)$ (completamente) diferente do anterior;
3) Não devem existir duas entradas m_1 e m_2 tal que $H(m_1) = H(m_2)$.

Pelas características citadas acima é fácil perceber que quanto maior a cadeia de *hash* calculada maior a segurança fornecida pelo algoritmo.

Existem diversos algoritmos que fornecem estas características atualmente, como MD4, MD5, SHA-1, SHA-256, SHA-512 e SHA-3. Podemos destacar atualmente o SHA-256, que faz parte da família SHA-2, que entre 2002 e 2012 foi a família de algoritmos padrão adotada pelo NIST (Instituto Nacional de Padrões e Tecnologia, da Câmara de Comércio dos EUA) [12]. Por esta razão se tornou um algoritmo muito utilizado. A partir de 2012 NIST substituiu o padrão SHA-2 pelo SHA-3, que é uma adaptação do método de *hash* conhecido como Keccak [13].

Já os algoritmos de criptografia são definidos em [11] como algoritmos que recebem uma entrada m, e de posse de uma chave criptográfica K_A calculam uma cadeia $K_A(m)$. Da mesma forma, esses algoritmos, de posse de uma chave K_B, conseguem calcular de volta a cadeira original m tal que $K_B(K_A(m)) = m$. Sendo assim esses algoritmos conseguem transformar a mensagem em um formato que não é "legível" por quem não sabe o algoritmo de encriptação utilizado ou não possua nenhuma das chaves criptográficas K_A e K_B.

Existem algoritmos em que K_A e K_B são iguais, conhecidos como algoritmos de chaves simétricas, e algoritmos em que K_A e K_B são diferentes, conhecidos como algoritmos de chaves assimétricas. Estes últimos se destacam por serem utilizados em um esquema conhecido como chaves pública e privada. A chave K_B, de decrepitação, é pública, ou seja, conhecida por todos, e K_A, de encriptação, é privada, ou seja, conhecida apenas pelo seu proprietário. O nível de segurança deste tipo de algoritmo está na dificuldade em se descobrir K_A, privada, a partir de K_B, pública.

Nos métodos de assinatura, o remetente, de posse do *hash* da mensagem que será enviada, pode encriptá-lo utilizando a sua chave privada. Esta cadeia de caracteres, que é o resultado do *hash* encriptado pela chave do remetente, é conhecida como assinatura. O receptor recebe a mensagem e a assinatura, e utilizando a chave pública do remente a decripta, obtendo

o *hash* da mensagem, que é então verificado com o *hash* calculado por ele mesmo. Sendo assim ele tem uma segurança de que a mensagem foi enviada por quem diz ter enviado, pois ela só pode ter sido encriptada com a chave privada dele, que em tese, só ele possui [11].

Esta forma de assinatura, encriptando apenas o *hash*, é utilizada por que o processo de encriptação é muito custoso, tornando a encriptação de uma mensagem grande inviável.

Entre os algoritmos de criptografia de chave pública e privada se destacam o RSA (*Rivest-Shamir-Adleman*) [7] e o ECC (*Elliptic Curve Cryptography*) [8]. O RSA gera pares de chaves privadas e públicas se utilizando de fatoração de números primos, e tem sua segurança baseada na dificuldade computacional disto [11]. Já o ECC se baseia em propriedades de curvas elípticas. O ECC promete ser mais rápido pois consegue trazer o mesmo nível de segurança que o RSA utilizando chaves bem menores. A Tabela I, traduzida de [14], mostra a comparação dos níveis de segurança obtidos com diferentes tamanhos de chaves.

Tabela I
COMPARAÇÃO ENTRE OS NÍVEIS DE SEGURANÇA DE ACORDO COM OS TAMANHOS DE CHAVES ENTRE O RSA E O ECC [14]

Nível de segurança	**Tamanho de chave RSA**	**Tamanho de chave ECC**	**Curvas ECC**
80	1024 bits	160-223 bits	secp192r1, secp192k1
112	2048 bits	224-255 bits	secp224r1, secp224k1
128	3072 bits	256-383 bits	secp256r1, secp 256k1, bp256r1
192	7680 bits	384-511 bits	secp384r1, bp384r1
256	15.360 bits	512+ bits	secp521r1, bp521r1

C. Plataforma ODROID-XU3

O ODROID-XU3 [15] é uma plataforma multi-core heterogênea (HMP - *Heterogeneous Multi-Processing*) baseada no Samsung Exynos 5422 SoC com tecnologia ARM® big.LITTLE™, conforme ilustrado na Figura 1. Para processamento, estão disponíveis 8 processadores, sendo 4 núcleos Cortex-A15 (2.0 GHz) e 4 núcleos Cortex-A7 (1.4 GHz). Dispõe ainda de 2GB de memória LPDDR3 933MHz, entradas USB 2.0/3.0, HDMI 1080p e rede Ethernet 10/100 Mbps. Além disso, possui um mecanismo integrado de monitoramento com sensores que informam frequência, voltagem, amperagem e consumo de energia.

Além do baixo consumo de energia, uma das vantagens do componente é o suporte *open source*, que permite executar Android™ 4.4 e Ubuntu 15.04.

O seu sucessor é o Odroid-XU4, é mais compacto, porém com hardware semelhante, compatível ao XU3, e suporte ao Ubuntu 16.04 e Android™ 7.1.

III. TRABALHOS RELACIONADOS

O trabalho de [16] avaliou o consumo energético e a vazão para comunicações seguras com RSA e ECC em dispositivos IoT no contexto de HTTPS (Hypertext Transport Protocol Secure) e TLS (Transport Layer Security). Dois cenários foram simulados: Comunicação entre um dispositivo de ponta e um gateway, e comunicação direto entre dois dispositivos de ponta. No primeiro cenário os resultados indicam que o ECC é mais eficiente que o RSA tanto energeticamente quanto em vazão, mas isto depende de qual curva está sendo utilizada. No trabalho a que apresentou o melhor resultado foi a secp256r1. Já na comunicação direta os consumos energéticos foram cerca de três vezes maiores com vazões cerca de cinco vezes menores, isto devido aos delays na comunicação.

Em [17], os algoritmos de *hash* e criptografia também são avaliados em um contexto de IoT por meio de testes nos dispositivos físicos. Este trabalho difere dos anteriores por utilizar um simulador, o que flexibiliza o *framework* desenvolvido, permitindo estendê-lo mais facilmente para outras arquiteturas e dispositivos. Outras diferenças estão nos dispositivos utilizados. Enquanto em [17] são utilizados um Arduino Uno R3 e um ESP32 como representantes da categoria de dispositivos IoT, este trabalho foca no ODROID-XU3, que possui o processador Samsung Exynos 5422 SoC que ao utilizar tecnologia ARM® big.LITTLE™ permite comparar dois tipos de *cores*, um focado em desempenho (A15) e outro focado em eficiência energética (A7).

Em [18] foi realizada uma análise comparativa entre RSA e ECC com diferentes níveis de segurança utilizando o Matlab e um computador com Intel Pentium dual-core (1.6GHz e 1MB L2) com 2GB DDR2. Os resultados indicam que o ECC é mais eficiente que o RSA em tempo de execução, sendo mais adequado para dispositivos com restrição de recursos. Similarmente, em [19] o desempenho do RSA e do ECC é comparado por meio de *benchmarks* executados em um computador com processador *dual core* de 2.4GHz com 2GB de memória RAM. O trabalho proposto se diferencia destes últimos em relação à arquitetura e no contexto de recursos limitados dos sistemas embarcados utilizados em redes IoT. Além disto o enfoque deste trabalho se dá no consumo energético dos algoritmos, algo avaliado apenas em [17] entre os trabalhos citados.

IV. METODOLOGIA

O objetivo do trabalho é comparar o desempenho e consumo energético dos algoritmos de *hash* SHA-2 e SHA-3 de 256 bits, e entre os algoritmos de criptografia RSA, com chave de 2.048 bits, e o ECC com a curva SECP256K1, que utiliza chaves de 256 bits. Porém, as comparações entre estes são apenas algumas das que podem ser feitas para analisar o impacto das estratégias de comunicação nos diversos tipos de sistema. Portanto, foi tomado como objetivo secundário a criação de uma infraestrutura que facilitasse novos testes e *benchmarks*.

Fazer *benchmarks* em diferentes arquiteturas não é uma tarefa simples, pois seria necessário ter acesso a dispositivos com essas arquiteturas. Para evitar isso podemos nos abster de dados absolutos e usar simuladores de alto nível de abstração, passando a analisar os dados de forma relativa.

Entre os simuladores existentes se destaca o já citado Gem5 [4]. Ele faz simulações levando em consideração diversas características da arquitetura, como níveis e tamanho de *cache*,

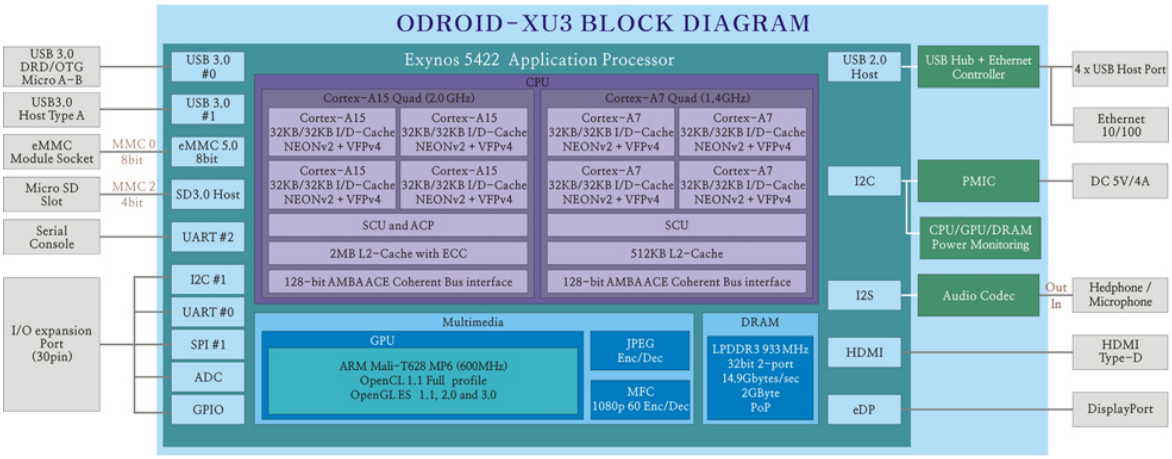

Figura 1. Diagrama de Blocos do ODROID-XU3 [15].

estratégias de coerência de memória, *pipelines*, etc. Isto o torna um ótimo candidato para fazer os *benchmarks* citados.

Para as estimativas de consumo energético foi escolhido o Gemstone [5], pela sua adaptação para utilização de simulações Gem5 nos modelos obtidos pelo processo descrito em [20]. A metodologia de [20], e adaptada em [5], utiliza métodos empíricos de modelagem e estimativa, o que leva a erros muito baixos entre os dados reais e os simulados. Em [20] o estudo de caso mostrou erros na casa 6,6%. No entanto este método dificulta a exploração de diferentes arquiteturas e dispositivos, pois é necessário que se obtenha os modelos destes, o que exige medidas e testes feitos diretamente nas plataformas físicas.

O GemStone se utiliza de simulações *full-system* do Gem5, ou seja, simulam todo o funcionamento da plataforma com sistema operacional, o que faz as simulações serem mais custosas, porém mais completas, pois simulam toda a pilha de componentes, desde o sistema operacional até a aplicação.

Outro fator determinante na escolha de utilização do Gemstone é a disponibilidade de um modelo já testado, verificado e validado, ou seja, pronto para uso. A plataforma modelada é a ODROID-XU3, dotada de um processador Samsung Exynos 5422, arquitetura ARM big.LITTLE, que possui 4 *cores* Cortex™–A15 e 4 *cores* Cortex™–A7. O processo de modelagem desta plataforma é mostrado em [20].

Para os *benchmarks* foram desenvolvidos dois testes principais: Remetente (*sender*), focado no envio de mensagens, e Destinatário (*receiver*), focado no recebimento de mensagens. Os fluxogramas da Figura 2 mostram a sequência de passos de cada um. É importante frisar que o passo de assinar a mensagem no teste de envio, e o de verificar assinatura no teste de recebimento, são ativados ou não por *flags* de compilação, não necessitando de instruções de verificação.

Os algoritmos sendo comparados são colocados como parte constituinte dos testes ao desempenharem suas funções nestes dois testes base. Desta forma, buscamos uma comparação justa, pois os fatores externos que podem influenciar são minimizados, já que as interfaces de uso são as mesmas em todos os casos.

Buscou-se para estes algoritmos implementações bem consolidadas. Entre as diversas implementações disponíveis as da biblioteca wolfSSL[1] se evidenciam. Trata-se de uma solução *open source*, bem estabelecida e muito utilizada pela comunidade, possui documentação extensiva e foi desenvolvida pensando em sistemas com recursos mais restritos.

Todo o código-fonte dos *benchmarks*, com *scripts* de compilação e simulação, está disponível no repositório do endereço https://gitlab.com/ppgcomp-gem5/crypto-profiler.

Foram excluídos dos testes os passos de geração e trocas de chaves. Desta forma, considera-se que o remetente e o destinatário possuem pares de chaves pública e privada pré-definidas. A comparação de desempenho e consumo energético é voltada à capacidade dos algoritmos na execução de suas funções principais: calcular um *hash*, para o SHA-2 e SHA-3, e criptografar e decriptar mensagens para o RSA e o ECC.

No caso dos *benchmarks* com assinatura, temos 4 combinações: SHA-2 com ECC, SHA-3 com ECC, SHA-2 com RSA e SHA-3 com RSA.

Todos os *benchmarks* são feitos com uma mesma mensagem de entrada:*"{'data': 'Isto é um exemplo de payload!'}"*, que foi pré-serializada utilizando *Protocol Buffers* (*protobuf*)[2], muito utilizado em sistemas embarcados. A entrada então se tornou um buffer com 21 bytes de tamanho.

Note que o *hash* do SHA-3 utilizado é o de 256 bits, para ter uma certa similaridade com o SHA-2 também de 256 bits, e assim termos uma melhor comparação de desempenho, com uma menor interferência do tamanho do *hash*.

Já para os algoritmos de assinatura, foi escolhido o ECC com o SECP256K1 pois esta é uma curva que se popularizou após a publicação do Bitcoin, que a utiliza para assinatura das comunicações com o seu Blockchain. Esta curva utiliza chaves de 256 bits e para seguir o mesmo padrão do *hash*, onde buscamos uma similaridade de nível de segurança, e utilizando a Tabela I de comparação de tamanho das chaves com o nível

[1]https://www.wolfssl.com

[2]https://developers.google.com/protocol-buffers

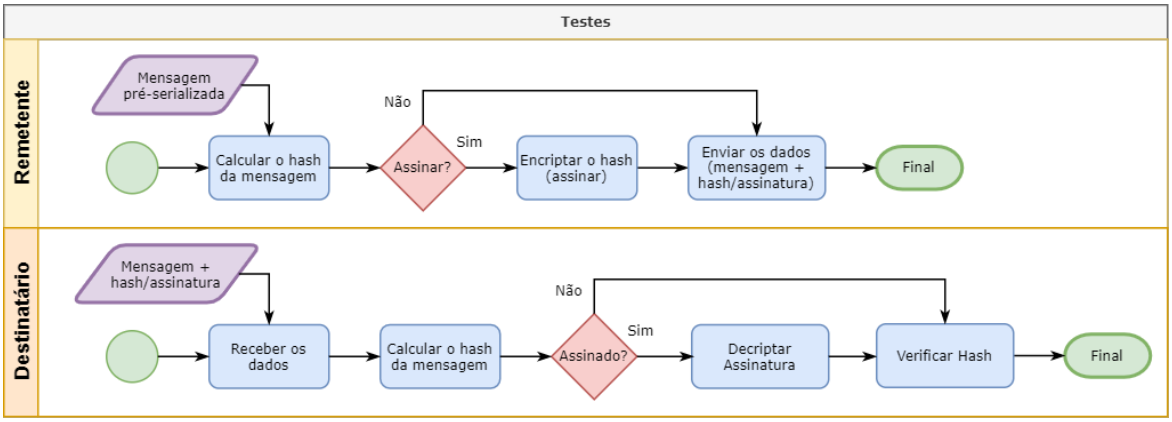

Figura 2. Fluxograma dos testes realizados no remetente e receptor.

de segurança oferecido entre RSA e ECC presente na Seção II, foram geradas chaves de 2.048 bits para o RSA, ficando assim ambos no limiar entre os níveis de segurança 112 e 128.

Para as simulações todos os executáveis são colocados em uma imagem do sistema operacional Linux disponibilizada pelo Gemstone. Cada simulação inicializa o sistema operacional e executa um dos cenários do *benchmark*. Destaca-se que a inicialização do sistema operacional não influencia nas estatísticas de execução obtidas, pois elas são zeradas imediatamente antes de cada *benchmark* e salvas imediatamente após a finalização.

O processo de modelagem da plataforma ODROID-XU3 se utilizou do recurso DVFS (*Dynamic Voltage and Frequency Scale*) da arquitetura ARM® big.LITTLE™. Com ele é possível variar a tensão de alimentação e a frequência de *clock* dos *cores* de formar a buscar o melhor balanceamento entre tempo de processamento e consumo energético. Sendo assim a estimativa do consumo de energia é feita com os *benchmarks* executando nos *cores* A15 nas frequências de 600MHz, 1GHz, 1.4GHz e 1.8GHz, e também nos *cores* A7 nas frequências de 200MHz, 600MHz, 1GHz, e 1.4GHz. A Tabela II mostra a tensão para cada frequência em cada *core*.

Embora a frequência máxima da plataforma seja 2GHz, a modelagem de energia feita em [20] limitou-se a 1.8GHz, pois com 2GHz os controles térmicos da placa eram ativados. Mesmo com 1.8GHz, um intervalo era dado entre cada um dos testes para resfriar o processador.

O Gem5 não permite de forma padrão a variação da tensão de alimentação dos *cores*, desta forma, parte da adaptação do Gemstone é permitir inserir estas tensões para cada frequência no momento da estimativa do consumo energético.

O compilador utilizado para compilar os *benchmarks* foi o arm-linux-gnueabihf-gcc, que já utiliza as operações em ponto flutuante implementadas em *hardware*. As bibliotecas necessárias (*wolfssl* e *math*) são inseridas de forma estática (*flag –static*). Nenhuma otimização de compilador foi ativada.

O cálculo de estimativa de energia leva em consideração os processos executando em cada um dos 4 *cores*, sejam os A15 ou os A7. Como os *benchmarks* criados não possuem

Tabela II
TENSÕES DE ALIMENTAÇÃO PARA CADA *core* EM CADA FREQUÊNCIA.

Frequência	Tensão A7	Tensão A15
200MHz	0,9195V	–
600MHz	0,9440V	0,9120V
1000MHz	1.8900V	0,9540V
1400MHz	1.2530V	1,0420V
1800MHz	–	1,1830V

processos paralelos, e para evitar que processos do sistema operacional executando nos outros *cores* afetem os dados obtidos, foi escolhido lançar em cada simulação 4 processos de cada *benchmark*, cada um rodando em um *core* do Cortex™ em questão.

As simulações do Gem5 são consistentes independente da máquina ou sistema operacional que as estejam executando, ou seja, geram sempre os mesmos resultados a cada execução. Desta forma, não foi necessário considerar mais de uma execução para calcular a média ou outros dados estatísticos.

Ao final de todo o processo de simulação do Gem5 e procedimentos do Gemstone é obtido um arquivo com valores separados por vírgula (CSV) com os todos os dados gerados pelo Gem5, junto com a aproximação da potência média (em W) feita pelo Gemstone, para cada *benchmark*. Desta maneira, a estimativa de consumo energético (em J) é feita multiplicando a potência pelo tempo de execução dos *benchmarks*.

V. RESULTADOS OBTIDOS

A análise se inicia pelos *benchmarks* que não possuem assinatura, ou seja, possuem apenas cálculo de *hash*.

A Tabela III sumariza os dados de tempo de execução (em segundos) e energia consumida (em Joules) obtidos, agrupando-os por algoritmo, *core* e frequência de *clock*, calculando ainda um total, ao somar os dados relacionados de remetente e destinatário.

Um esquema de cores foi aplicado nos totais de energia consumida calculados, de forma a facilitar a identificação dos valores em uma escala relativa, partindo do verde, para valores baixos, seguindo para amarelo e, por fim, vermelho conforme os valores vão aumentando em relação aos seus pares.

Tabela III
DADOS DE TEMPO DE EXECUÇÃO (EM SEGUNDOS) E CONSUMO ENERGÉTICO (EM JOULES) DOS *benchmarks* DE SHA-2 E SHA-3, SEM ASSINATURA.

			200MHz		600MHz		1000MHz		1400MHz		1800MHz	
			Tempo	Energia	Tempo	Energia	Tempo	Energia	Tempo	Energia	Tempo	Energia
SHA-256	A7	Remetente	0.00999	0.00023	0.01009	0.00037	0.00615	0.00077	0.00447	0.00135		
		Destinatário	0.00834	0.00014	0.00967	0.00042	0.00595	0.00087	0.00512	0.00153		
		Total	0.01833	0.00037	0.01976	0.0008	0.0121	0.00163	0.00959	0.00288		
	A15	Remetente			0.00929	0.00378	0.00683	0.00462	0.01277	0.01215	0.00447	0.01043
		Destinatário			0.01044	0.00525	0.00616	0.00455	0.0107	0.01351	0.0038	0.00813
		Total			0.01973	0.00903	0.01298	0.00917	0.02347	0.02566	0.00827	0.01856
SHA-3 256	A7	Remetente	0.01038	0.00025	0.01023	0.00038	0.00588	0.00077	0.00461	0.0013		
		Destinatário	0.0139	0.00034	0.01013	0.0004	0.00652	0.00091	0.00454	0.00146		
		Total	0.02427	0.00059	0.02037	0.00077	0.0124	0.00168	0.00915	0.00276		
	A15	Remetente			0.01027	0.00526	0.00688	0.0047	0.01246	0.01561	0.00449	0.00835
		Destinatário			0.00833	0.00429	0.00635	0.00437	0.01048	0.00858	0.00461	0.01012
		Total			0.0186	0.00955	0.01323	0.00908	0.02294	0.02419	0.0091	0.01846

Ainda na Tabela III, nos totais de tempo de execução, foi aplicado um esquema de barra, com o mesmo objetivo de facilitar a identificação dos valores em uma escala relativa. A barra preenche proporcionalmente a célula de acordo com o valor em relação aos seus respectivos.

A Figura 3 mostra os dados totais de uma maneira gráfica, colocando os pontos de tempo de execução com seus respectivos consumos energéticos.

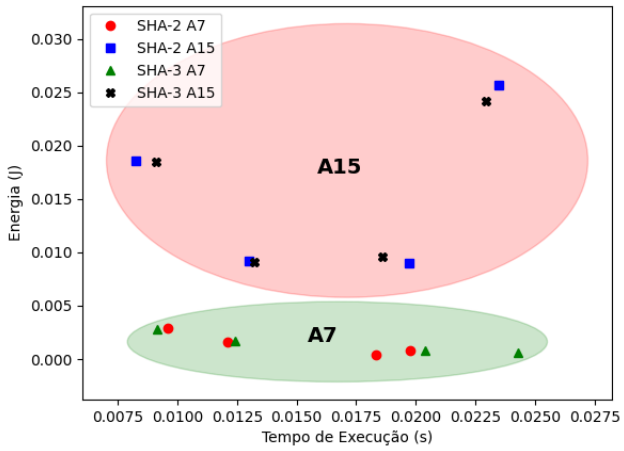

Figura 3. Tempo de execução x Energia consumida pelos *benchmarks* de SHA-2 e SHA-3 sem assinatura.

É possível observar que tanto o SHA-2 quanto o SHA-3 possuem um melhor comportamento energético quando executados nos *cores* A7, independente da frequência de *clock*. Os tempos de execução desses algoritmos variam muito pouco em relação ao *core* em que estão sendo executados, com a diferença no consumo de energia se justificando pela menor potência consumida pelos *cores* A7, que possuem menos recursos, como *cache* e *pipeline*. A Figura 1 mostra algumas das diferenças entre os *cores*.

Analisando apenas os dados do *core* A7, quanto maior a frequência, maior é o consumo de energia, mesmo com a diminuição no tempo de execução. Isto se deve ao fato de a frequência mais alta necessitar de uma maior tensão de alimentação (Tabela II), que possui um efeito quadrático na potência consumida pelos *cores*. Entretanto, observa-se na Figura 3 que a melhor eficiência acontece no SHA-3 com a frequência de 1.4GHz, sendo o ponto mais próximo da origem.

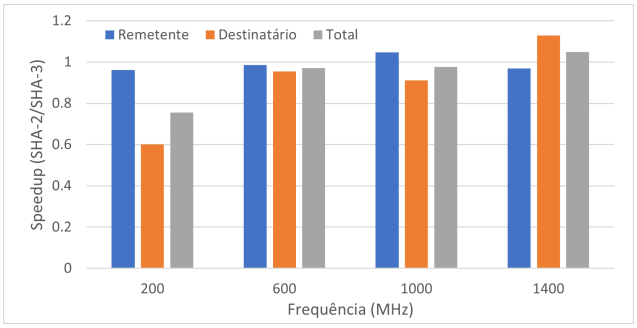

Figura 4. Tempos de execução relativos dos *benchmarks* de SHA-2 e SHA-3 sem assinatura.

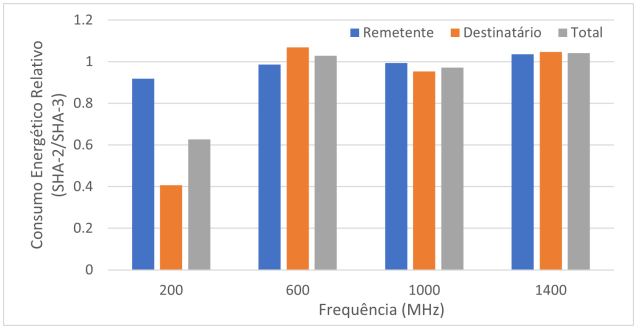

Figura 5. Energia consumida relativa dos *benchmarks* de SHA-2 e SHA-3 sem assinatura.

Por meio dos gráficos das Figuras 4 e 5, que mostram os dados relativos de tempo de execução e consumo energético entre o SHA-2 e o SHA-3, pode-se observar que o SHA-2

possui um comportamento energético visivelmente superior apenas no *clock* de 200 MHz, executando cerca de 25% mais rápido, e um consumo energético aproximadamente 38% menor. Nas outras frequências, tanto o tempo de execução quanto o consumo energético são muito semelhantes. Devido a superioridade na segurança fornecida pelo SHA-3, motiva-se o seu uso em detrimento do SHA-2, com exceção do caso de baixa frequência de *clock*, onde a diferença entre ambos se tornou significativa, indicando o uso do SHA-2.

A Tabela IV mostra os dados de quando a assinatura é ativada com RSA ou ECC. Os testes foram feitos com o SHA-3 como algoritmo que gerou o *hash* posteriormente encriptado. O esquema de cores e barras é o mesmo da tabela anterior.

A Figura 6 mostra os dados totais da Tabela IV de uma maneira gráfica, colocando os pontos de tempo de execução com seus respectivos consumos energéticos.

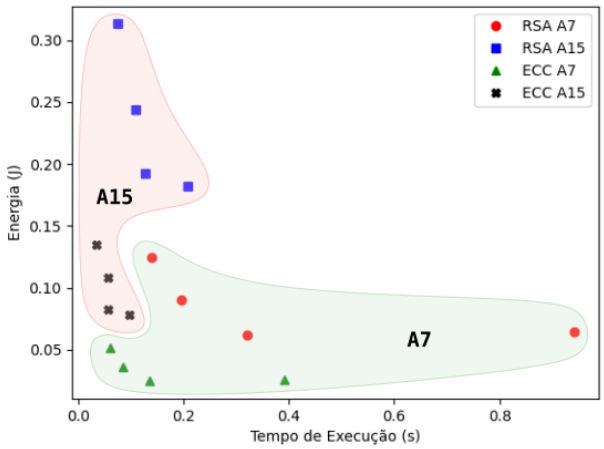

Figura 6. Tempo de execução x Energia consumida pelos *benchmarks* SHA-3 com assinatura RSA e ECC.

A Tabela IV e a Figura 6 mostram que, assim como os *hashes* SHA-2 e SHA-3, tanto o RSA quanto o ECC possuem um melhor comportamento energético quando executados no *core* simplificado A7. Porém, diferentemente dos *hashes*, a execução nos *cores* A15 é mais rápida, contudo não o suficiente para compensar a sua potência maior. Por exemplo, com 600MHz, as execuções no *core* A15 são cerca de 30% mais rápidas, porém a energia consumida é cerca de três vezes maior. Conforme aumenta-se a frequência, o tempo de execução vai se igualando e a diferença no consumo de energia diminuindo, porém continua em torno de duas vezes maior.

A mesma tendência dos *hashes* também acontece em relação às frequências de *clock*: quanto maior a frequência, maior a energia consumida na execução. E a mesma justificativa dos *hashes* cabe aqui: a influência da tensão de alimentação maior retira qualquer vantagem do tempo de execução menor.

Pelo gráfico da Figura 6, analisando um balanceamento entre tempo de execução *versus* energia consumida, conclui-se que a melhor eficiência acontece no ECC com frequência de 1.400MHz, pois é o ponto mais próximo da origem, seguido de forma muito próxima pelo ECC com frequência de 1.000MHz.

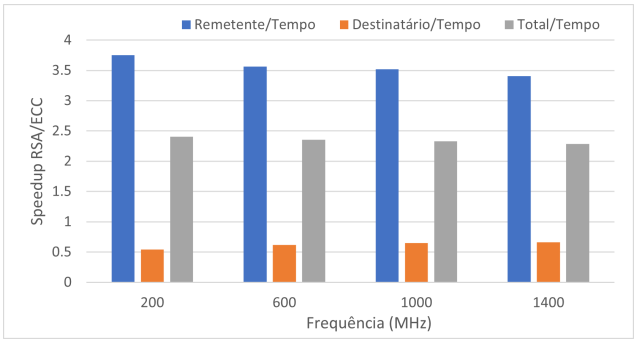

Figura 7. Tempos de execução relativos dos *benchmarks* com assinatura RSA e ECC.

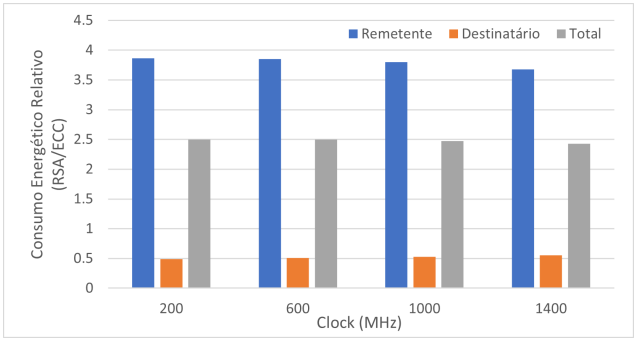

Figura 8. Energia consumida relativa dos *benchmarks* com assinatura RSA e ECC.

Por meio dos gráficos das Figuras 7 e 8, que mostram os dados relativos de tempo de execução e consumo energético entre o ECC e o RSA, pode-se observar que o ECC possui um comportamento bem superior ao do RSA, tanto no tempo de execução quanto no consumo energético. Ambos são cerca de 2,5 vezes maiores no RSA que no ECC, favorecendo muito o uso deste.

Entretanto, é importante observar que a superioridade do ECC reside na sua encriptação (*benchmarks* remetentes), já que na verificação de assinaturas (*benchmarks* destinatários) o RSA apresentou comportamento superior. Portanto, podem existir cenários em que o RSA pode se mostrar um melhor candidato que o ECC, como nos que as mensagens de *broadcasts* representam a grande maioria das comunicações.

VI. Conclusão e Trabalhos Futuros

Este trabalho mostrou como os algoritmos de *hash* e assinatura estudados possuem tanto desempenho como consumo energético superiores quando executados nos *cores* A7, que possuem arquitetura mais simplificada.

Nesses *cores* os algoritmos SHA-2 e SHA-3 possuem um desempenho e um consumo energético muito semelhante, favorecendo o uso do SHA-3 por sua maior segurança. A exceção à esta conclusão é o caso de frequência mais baixa (200MHz) onde o SHA-2 teve desempenho e consumo energético perceptivelmente superiores.

Tabela IV
TEMPO DE EXECUÇÃO (EM SEGUNDOS) E CONSUMO ENERGÉTICO (EM JOULES) DO *benchmark* SHA-3 COM ASSINATURA RSA E ECC.

			200MHz		600MHz		1000MHz		1400MHz		1800MHz	
			Tempo (s)	Energia (J)	Tempo (s)	Energia (J)	Tempo (s)	Energia (J)	Tempo (s)	Energia (J)	Tempo (s)	Energia (J)
RSA	A7	Remetente	0.85261	0.059505	0.28649	0.0566729	0.17396	0.0825065	0.12396	0.1134234		
		Destinatário	0.08976	0.0051431	0.03474	0.0051131	0.02279	0.0078319	0.01678	0.0113962		
		Total	0.94237	0.0646481	0.32122	0.061786	0.19676	0.0903384	0.14074	0.1248196		
	A15	Remetente			0.18301	0.1633365	0.11092	0.1726468	0.08728	0.2149661	0.06297	0.2783443
		Destinatário			0.02556	0.0187078	0.01754	0.0196577	0.02171	0.028608	0.01221	0.0346918
		Total			0.20857	0.1820443	0.12846	0.1923045	0.109	0.2435741	0.07518	0.3130361
ECC	A7	Remetente	0.22728	0.0153941	0.08045	0.0147183	0.04946	0.0217235	0.03641	0.0308585		
		Destinatário	0.16488	0.0104616	0.05606	0.0099994	0.03496	0.0147883	0.02524	0.0205141		
		Total	0.39216	0.0258557	0.13651	0.0247176	0.08442	0.0365118	0.06165	0.0513726		
	A15	Remetente			0.05583	0.0465182	0.03237	0.0478137	0.03206	0.0622379	0.01966	0.0775011
		Destinatário			0.04169	0.0318034	0.0251	0.0349872	0.02419	0.0460382	0.01444	0.0576089
		Total			0.09752	0.0783215	0.05747	0.0828008	0.05625	0.108276	0.03411	0.13511

No caso dos algoritmos de assinatura tanto o ECC quanto o RSA tiveram tempo de execução menores nos *cores* A15, porém o consumo de energia é menor nos *cores* A7.

Comparando um método ao outro, o ECC se mostrou melhor no contexto dos remetentes, mas pior no contexto dos destinatários. Somando os dois casos o ECC continua superior, justificando a sua escolha, mas dependendo do contexto da aplicação o RSA pode ser a melhor opção, como nos casos onde as mensagens de *broadcast* representam a maior parte das comunicações.

Como trabalhos futuros pretende-se realizar a comparação de outros algoritmos de *hash* e assinatura, e também estender a análise de dados para compor uso da memória.

REFERÊNCIAS

[1] B. Jovanović, "Internet of Things statistics for 2021 - Taking Things Apart," DataProt, Mar. 21, 2021. [Online]. Available: https://dataprot.net/statistics/iot-statistics/

[2] M. Farooq, M. Waseem, S. Mazhar, A. Khairi, and T. Kamal, "A Review on Internet of Things (IoT)," *International Journal of Computer Applications*, vol. 113, pp. 1–7, 03 2015.

[3] S. Vashi, J. Ram, J. Modi, S. Verma, and C. Prakash, "Internet of Things (IoT): A vision, architectural elements, and security issues," in *2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, 2017, pp. 492–496.

[4] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, "The Gem5 Simulator," *SIGARCH Comput. Archit. News*, vol. 39, no. 2, p. 1–7, Aug. 2011. [Online]. Available: https://doi.org/10.1145/2024716.2024718

[5] M. Walker, S. Bischoff, S. Diestelhorst, G. Merrett, and B. Al-Hashimi, "Hardware-validated CPU performance and energy modelling," in *2018 IEEE International Symposium on Performance Analysis of Systems and Software (02/04/18 - 03/04/18)*, April 2018. [Online]. Available: https://eprints.soton.ac.uk/418538/

[6] National Institute of Standards and Technology, *FIPS PUB 202: SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*. pub-NIST, 2015.

[7] R. A. Mollin, *RSA and Public-Key Cryptography*. USA: CRC Press, Inc., 2002.

[8] D. Hankerson, A. J. Menezes, and S. Vanstone, *Guide to Elliptic Curve Cryptography*. Berlin, Heidelberg: Springer-Verlag, 2003.

[9] A. Pastório, L. Rodrigues, and E. de Camargo, "Uma Revisão Sistemática da Literatura Sobre Tolerância a Falhas em Internet das Coisas," in *Anais Estendidos do X Simpósio Brasileiro de Engenharia de Sistemas Computacionais*. Porto Alegre, RS, Brasil: SBC, 2020, pp. 57–64. [Online]. Available: https://sol.sbc.org.br/index.php/sbesc_estendido/article/view/13091

[10] R. Mahmoud, T. Yousuf, F. Aloul, and I. Zualkernan, "Internet of things (IoT) security: Current status, challenges and prospective measures," in *2015 10th International Conference for Internet Technology and Secured Transactions (ICITST)*, 2015, pp. 336–341.

[11] J. F. Kurose and K. W. Ross, *Redes de Computadores e a Internet: Uma abordagem top-down*, 3rd ed. São Paulo: Addison Wesley, 2006.

[12] National Institute of Standards and Technology, *FIPS PUB 180-2: Secure Hash Standard*. pub-NIST: pub-NIST, 08 2002, supersedes FIPS 180-1 1995 Apr 17. [Online]. Available: https://csrc.nist.gov/publications/detail/fips/180/2/archive/2002-08-01

[13] ——, *FIPS 180-4: Secure Hash Standard*. pub-NIST: pub-NIST, 08 2015. [Online]. Available: https://csrc.nist.gov/publications/detail/fips/180/4/final

[14] M. Suárez-Albela, P. Fraga-Lamas, and T. M. Fernández-Caramés, "A Practical Evaluation on RSA and ECC-Based Cipher Suites for IoT High-Security Energy-Efficient Fog and Mist Computing Devices," *Sensors*, vol. 18, no. 11, 2018. [Online]. Available: https://www.mdpi.com/1424-8220/18/11/3868

[15] J.-L. Aufranc, "Hardkernel unveils $179 odroid-xu3 development board powered by samsung exynos 5422 soc," CNXSoft, Jul. 14, 2014 [Online]. [Online]. Available: https://www.cnx-software.com/2014/07/08/hardkernel-unveils-179-odroid-xu3-development-board-powered-by-samsung-exynos-5422-soc/

[16] M. Suárez-Albela, P. Fraga-Lamas, and T. M. Fernández-Caramés, "A Practical Evaluation on RSA and ECC-Based Cipher Suites for IoT High-Security Energy-Efficient Fog and Mist Computing Devices," *Sensors*, vol. 18, no. 11, 2018. [Online]. Available: https://www.mdpi.com/1424-8220/18/11/3868

[17] R. Albarello, M. Oyamada, and E. de Camargo, "Avaliação de Algoritmos de Criptografia e Implementação de um Protocolo Leve para Comunicação entre Dispositivos IoT," in *Anais Estendidos do X Simpósio Brasileiro de Engenharia de Sistemas Computacionais*. Porto Alegre, RS, Brasil: SBC, 2020, pp. 65–72. [Online]. Available: https://sol.sbc.org.br/index.php/sbesc_estendido/article/view/13092

[18] D. Mahto and D. YADAV, "RSA and ECC: A comparative analysis," *International Journal of Applied Engineering Research*, vol. 12, pp. 9053–9061, 01 2017.

[19] N. J. G. SAHO and E. C. Ezin, "Comparative Study on the Performance of Elliptic Curve Cryptography Algorithms with Cryptography through RSA Algorithm," in *CARI 2020 - Colloque Africain sur la Recherche en Informatique et en Mathématiques Apliquées*, Thiès, Senegal, Oct. 2020. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02926106

[20] M. Walker, S. Bischoff, S. Diestelhorst, G. Merrett, and B. Al-Hashimi, "Hardware-Validated CPU Performance and Energy Modelling," in *2018 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2018, pp. 44–53.