Uma avaliação de *toolkits* para criptografia baseada em emparelhamento bilinear

Frederico M. B. Sampaio[1], Lucas Goulart Grossi[1], Antonio L. Maia Neto[1], Ítalo Cunha[1], Antonio A. F. Loureiro[1], Leonardo B. Oliveira[1]

[1]Universidade Federal de Minas Gerais (UFMG) Belo Horizonte – MG – Brasil

Abstract. *Bilinear pairing has many applications in cryptography. Several ID-based, and also attribute-based cryptosystems were proposed using bilinear pairings of algebraic curves. This work aims at presenting the principal pairing-based cryptography toolkits, comparing their main characteristics as optimization, readability and coverage. This paper also contemplates an evaluation of the performance and energy efficiency of each of them and their crypto-primitives, in order to define the pros and cons of each toolkit.*

1. Introdução

O emparelhamento bilinear se tornou um instrumento para a criação de novos criptossistemas [Sakai et al. 2000]. Provavelmente, sua aplicação de maior sucesso é a IBE – *Identity-based Encryption* [Boneh and Franklin 2001]. O desenvolvimento de *softwares* criptográficos exige uma base que forneça abstração, estabilidade, produtividade e tenha desempenho viável para sua aplicação prática. Essa viabilidade é ainda mais crítica em aplicações de IoT – *Internet of Things* [Atzori et al. 2010]. Contudo, o emparelhamento bilinear é um tema complexo, pois envolve muitos fundamentos matemáticos. Essa complexidade se reflete nos algoritmos, dada a dificuldade de construção e implementação eficientes. Por isso, os *toolkits* que implementam a primitiva do emparelhamento bilinear são tão importantes.

Objetivo: O objetivo deste trabalho é pesquisar sobre *toolkits* criptográficos e sua adequação para a construção de *softwares* criptográficos baseados em emparelhamento bilinear, como esquemas de IBE, incluindo o ABE – *Attribute Based Encryption* [Sahai and Waters 2005]. Essas tecnologias possibilitam diversas características relevantes, como o uso chaves públicas sem necessidade de certificados digitais, criptografia que preserva a privacidade das partes, autenticação baseada em atributos e varias outras aplicações de segurança.

O trabalho é composto de uma avaliação qualitativa de *toolkits* disponíveis publicamente, focando critérios relevantes para o desenvolvimento de *softwares*. Também foram elaborados experimentos para avaliação quantitativa do desempenho dos *softwares* avaliados. Alguns *toolkits* já possuem programas de *benchmark*. Mesmo assim, foram desenvolvidos outros programas para fornecer uma oportunidade de uso prático dos diversos *toolkits*.

2. Avaliação Qualitativa

O trabalho teve início com uma pesquisa de *softwares* disponíveis. Os três *toolkits* de emparelhamento em curvas elípticas mais conhecidos são o MIRACL[1], o PBC[2] e o RELIC[3] . Existem vários outros *softwares* que provêm emparelhamento. Porém, quase todos são baseados em um desses *toolkits* já citados. Uma avaliação qualitativa é apresentada na Tabela 1, onde fazemos uma relação entre as métricas que consideramos mais importantes para essa análise e os *toolkits* que são avaliados neste trabalho.

O MIRACL é o *software* que possui a API mais complexa dentre os *toolkits* analisados, principalmente na parte de inicialização/configuração. Existem diversos exemplos de código, mas eles dependem de amplo conhecimento, incluindo fundamentos matemáticos na área de curvas elípticas e emparelhamento. Isso reduz o poder de abstração e dificulta muito sua curva de aprendizagem. Um destaque no MIRACL é o uso de classes e sobrecarga de operadores da Linguagem C++ para tipos e operações algébricas relacionadas com inteiros de precisão arbitrária e curvas elípticas. Isso amplia a capacidade de leitura e manutenção. Todavia, isso não facilita o uso da API.

O PBC provavelmente foi o primeiro *software open-source* específico para emparelhamentos baseado em curvas elípticas e, provavelmente, o mais conhecido e utilizado. Sua API é a mais simples, genérica e produtiva dentre as bibliotecas analisadas. Seu principal destaque é a abstração e generalização: o mesmo código funciona para todos os tipos de curvas suportados pelo *toolkit*, bastando uma configuração inicial simples, via arquivo texto. Dos três principais *softwares* de emparelhamento estudados, o PBC é o único focado especificamente nesta área, inexistindo implementações comuns em criptossistemas como o AES ou o ECDSA.

O RELIC é voltado para aplicações IoT. Assim como no caso do MIRACL, nele não há dependências externas e sua utilização é adequada para diversos sistemas criptográficos modernos, em especial os baseados em curvas elípticas. Seu desenvolvimento é ativo, com vários recursos, otimizações e atualizações recentes disponíveis no repositório público do Github. O destaque do RELIC é seu equilíbrio. O código não é genérico como no PBC, porém possui boa abstração, semântica e legibilidade. Tendo como foco principal IoT, seu *design* busca minimizar o código gerado. Apresenta diversos protocolos criptográficos e, no geral, o uso é simples e direto.

Na avaliação feita na Tabela 1 o *toolkit* RELIC teve o melhor resultado qualitativo, com 9 altos, 5 médios e nenhum baixo. O MIRACL teve o segundo melhor desempenho com 6 altos, 6 médios e 3 baixos. O pior desempenho qualitativo foi apresentado pelo PBC, com 6 altos, 1 médio e 8 baixos.

3. Experimentos

Os *toolkits* criptográficos considerados nesse trabalho foram avaliados em função do desempenho e eficiência energética. Foram considerados as seguintes primitivas e protocolos criptográficos: (1) **AES** com chave de 256 $bits$ e 16KB de dados; (2) **ECDSA** com curva NIST-P256; (3) ***Pairing*** com emparelhamento em curva BN-P256 e (4) ***Point-***

[1] `https://github.com/CertiVox/MIRACL`
[2] `http://crypto.stanford.edu/pbc`
[3] `https://github.com/relic-toolkit/relic`

Métrica	MIRACL	PBC	RELIC
Atividade do projeto (desenvolvimento)	baixo	baixo	alto
Maturidade e estabilidade do *toolkit*	alto	alto	alto
Eficiência de tempo (desempenho)	alto	baixo	alto
Eficiência de memória	alto	baixo	alto
Eficiência energética	médio	baixo	médio
Facilidade de programação com o *toolkit*	baixo	alto	médio
Legibilidade do código baseado no *toolkit*	médio	alto	alto
API intuitiva e ortogonal	baixo	alto	médio
Cobertura e usabilidade de protocolos	médio	baixo	alto
Adequação para IoT	alto	baixo	alto
Qualidade e cobertura de manual	alto	alto	médio
Independência de outros *softwares*	alto	baixo	alto
Programas extras e utilitários	médio	alto	médio
Software de testes e *benchmark*	médio	médio	alto
Otimização (asm, *hardware*, etc)	médio	baixo	médio

Tabela 1. Avaliação qualitativa dos *toolkits* analisados.

Mult com multiplicação de ponto por escalar aleatório de 256 $bits$ em curva NIST-P256.Neste artigo foi utilizado o telefone celular Nexus5, fabricado pela LG Eletronics cujas especificações são: processador ARMv7, modelo Qualcomm MSM 8974 de, no máximo, 2.27Ghz, 2GB de RAM e sistema operacional Android 5.1.1.

O OpenSSL[4] é um dos *toolkits* criptográficos mais utilizados. Por isso, ele é uma boa referência. Como não possui emparelhamento, os resultados dos experimentos incluem o OpenSSL apenas para estabelecer, quando possível, um ponto de comparação.

Durante a execução dos *toolkits*, foram colhidos dados de energia e tempo de execução de cada primitiva criptográfica isoladamente. Visando aumentar a confiabilidade dos resultados, realizamos vários testes para cada primitiva e, a partir destes, obtivemos a média dos gastos de tempo e energia. Os resultados de tempo e energia estão sintetizados na Tabela 2, onde o símbolo ✗ representa que a primitiva em questão não é contemplada pelo *toolkit*. Por questão de espaço optamos pela utilização desta tabela ao em vez de gráficos.

Os resultados da Tabela 2 sugerem uma relação intrínseca entre tempo e energia, visto que as primitivas mais lentas são também as de maior consumo energético. Tendo isso em vista, vamos realizar uma comparação unificada de desempenho e eficiência energética entre os *toolkit* e suas primitivas.

A primitiva mais onerosa é o *Pairing*, na qual o MIRACL e o RELIC possuem desempenho próximo e o PBC é mais de 4,5 vezes menos eficiente. Uma das primitivas menos onerosas, o *PointMult* é mais eficiente no OpenSSL, tendo desempenho melhor do que o MIRACL, PBC e RELIC em torno de 85%, 91% e 76%, respectivamente.

Analisando a primitiva ECDSA percebemos que suas três operações: gerar chave, assinar e verificar, são executadas de forma mais eficiente no OpenSSL, que tem desempenho similar ao RELIC nas duas primeiras e é cerca de 62% mais eficiente do que este na terceira operação. Comparado ao MIRACL, o OpenSSL é 90% mais eficiente ao gerar chave, 75% mais eficiente ao assinar e cerca de 80% mais eficiente ao verificar. O AES,

[4] `https://github.com/openssl/openssl`

apesar de conter duas etapas: cifrar e decifrar, é a primitiva menos onerosa. O desempenho de suas etapas é simétrico, sendo mais eficientes no OpenSSL. Quando comparado, o OpenSSL apresenta uma melhora de 30% para com o RELIC e 67% para com o MIRACL.

Algoritmo	Operação	Miracl	OpenSSL	PBC	Relic
AES	cifrar	0.98 / 1.80	0.24 / 0.60	✗	0.73 / 1.00
	decifrar	1.02 / 1.80	0.24 / 0.70	✗	0.95 / 1.00
	gerar chave	29.45 / 33.3	2.68 / 4.10	✗	3.64 / 4.00
ECDSA	assinar	15.30 / 17.1	3.19 / 5.00	✗	4.75 / 5.00
	verificar	23.31 / 27.0	3.79 / 5.40	✗	12.26 / 12.0
Pairing	executar	40.77 / 38.5	✗	181.0 / 213.0	40.16 / 41.1
PointMult	executar	13.85 / 17.1	2.02 / 3.80	22.6 / 23.10	8.30 / 9.20

Tabela 2. Tempo de execução (ms) / consumo energético (mJ).

4. Conclusões

O objetivo deste trabalho foi reunir os principais *toolkits* criptográficos baseados em emparelhamento bilinear e avaliá-los para auxiliar na decisão sobre seu uso no desenvolvimento de *softwares* aplicativos e em pesquisa. Por isso, além da avaliação quantitativa de desempenho de tempo e consumo energético, foram avaliados requisitos qualitativos como recursos, estabilidade, otimização e legibilidade.

O *toolkit* MIRACL tem muitos recursos, seu desempenho de tempo é adequado e seu *design* é viável para aplicações IoT. No entanto, sua API complexa, seu baixo poder de abstração, a difícil aprendizagem e o fato de ter registrado desempenho e eficiência energética menor do que a maioria dos *toolkits* avaliados, tornam sua utilização menos recomendada em alguns cenários. O mesmo se aplica ao PBC que, apesar de ser mais simples e genérico, é oneroso e não possui sistemas criptográficos comuns como o AES e o ECDSA.

O *toolkit* RELIC, devido principalmente ao seu *design* voltado para aplicações IoT e seu equilíbrio entre desempenho, consumo energético, recursos, abstração e legibilidade, se destaca como o *toolkit* mais adequado. Atualmente, o RELIC também é o *toolkit* com projeto de desenvolvimento mais ativo e atualizado.

Apesar de ter caráter referencial neste artigo, por não possuir emparelhamento, as comparações quantitativas também indicam o OpenSSL como uma boa opção, uma vez que é pelo menos 20% mais eficiente que os demais.

Referências

Atzori, L., Iera, A., and Morabito, G. (2010). The internet of things: A survey. *Computer Networks*.

Boneh, D. and Franklin, M. K. (2001). Identity-based encryption from the weil pairing. In *21st Annual Int'l Cryptology Conference on Advances in Cryptology*.

Sahai, A. and Waters, B. (2005). Fuzzy identity-based encryption. In *24th Annual Int'l Conference on Theory and Applications of Cryptographic Techniques*.

Sakai, R., Ohgishi, K., and Kasahara, M. (2000). Cryptosystems based on pairing. In *Symposium on Cryptography and Information Security (SCIS'00)*.

