Felipe de Azevedo Piovezan

ANÁLISE DA EFICIÊNCIA ENERGÉTICA DE ALGORITMOS DE CRIPTOGRAFIA BASEADOS EM CURVAS ELÍPTICAS

Tese submetida ao Curso de Bacharelado em Ciência da Computação para a obtenção do Grau de Bacharel em Ciência da Computação.
Orientador: Prof. Dr. Luiz Cláudio Villar dos Santos
Coorientador: Prof. Dr. Daniel Santana de Freitas

Florianópolis - Santa Catarina

2015

Ficha de identificação da obra elaborada pelo autor através do
Programa de Geração Automática da Biblioteca Universitária da UFSC.

A ficha de identificação é elaborada pelo próprio autor

Maiores informações em:
http://portalbu.ufsc.br/ficha

Felipe de Azevedo Piovezan

ANÁLISE DA EFICIÊNCIA ENERGÉTICA DE ALGORITMOS DE CRIPTOGRAFIA BASEADOS EM CURVAS ELÍPTICAS

Esta Tese foi julgada aprovada para a obtenção do Título de "Bacharel em Ciência da Computação", e aprovada em sua forma final pelo Curso de Bacharelado em Ciência da Computação.

Florianópolis - Santa Catarina, 02 de Dezembro 2015.

Prof. Dr. Renato Cislaghi
Coordenador

Banca Examinadora:

Prof. Dr. Daniel Santana de Freitas
Coorientador

Prof. Dr. Luiz Cláudio Villar dos Santos

Prof. Dr. José Luís Almada Güntzel

Prof. Dr. Daniel Santana de Freitas

AGRADECIMENTOS

TBC

“Nevermore”

The Raven

RESUMO

A criptografia de chaves públicas (PKC), ou criptografia assimétrica, uma das bases para a comunicação segura pela internet, fornece as primitivas para a troca de chaves, autenticação de usuários e assinatura digital. A maioria dos algoritmos de PKC usados atualmente são baseados em conjecturas de teoria dos números, como a dificuldade da fatoração de inteiros ou a dificuldade de alguma versão do logaritmo discreto. Este último, quando usado na versão de curvas elípticas, permite o uso de chaves e assinaturas menores para o mesmo nível de segurança de outros algoritmos, o que é interessante em ambientes com recursos limitados.

Durante o desenvolvimento dos algoritmos assimétricos, os principais nodos da rede eram grandes servidores e desktops. Recentemente, porém, a escala dos dispositivos que estabelecem comunicação segura pela internet está diminuindo consistentemente: dos laptops e celulares, até sensores e etiquetas RFID, todos necessitam de um canal seguro. Essa miniaturização, no entanto, veio acompanhada de restrições de processamento e de suprimento energético, o que conflita com o fato que algoritmos de PKC são computacionalmente caros: como, então, utilizá-los em ambientes com recursos limitados? O presente trabalho pretende, a partir de simulações, avaliar o gasto energético do sistema de memórias de processadores executando algoritmos de criptografia de curvas elípticas, bem como o impacto que técnicas de otimização têm na eficiência energética.

Palavras-chave: Palavras chave

ABSTRACT

The realm of public-key (asymmetric) cryptography (PKC) is considered to be one the pillars of secure communication over the internet, since it includes user authentication, key enchange and digital signatures. PKC algorithms in use are mostly based in certain number-theoretic conjectures, like the difficulty of integer factorization or the difficulty of some discrete-log problem. The latter, when used in the elliptic curve setting, allows for shorter keys and signatures while maintaining the same security level of other algorithms, which is interesting in a resource constrained environment.
During the development of the first asymmetric algorithms, most nodes in a network were either desktops or servers. Recently, however, the scale of the devices partaking in secure communations over the internet has been steadily decreasing: from laptops and cellphones to sensors and RFID tags, they all require a secure channel. This miniaturization is responsible for new restrictions on the processing power and battery supply available to each node, which directly conflicts with the fact that PKC algorithms are computationally expensive. What then is the best way to employ those algorithms in such restricted devices? Through the use of simulations, this work will evaluate the energy cost of the memory system of processors executing elliptic curve algorithms, as well as the impact that certain optimzation techniques have on energy efficiency.

Keywords: keywords

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

LISTA DE SÍMBOLOS

SUMÁRIO

1 INTRODUÇÃO

Com a expansão da internet, não apenas servidores e *desktops* passam a fazer parte da rede, mas também um grande número de pequenos dispositivos como PDAs, celulares e sensores. Acoplados aos processores de tais equipamentos, encontram-se transmissores de rádio que permitem a comunição com outros dispositivos, estejam eles em uma rede local ou à longa distância conectados por algum *gateway*. Alguns problemas em comum podem ser encontrados nesses dispositivos, como limitações na capacidade de processamento, quantidade de memória e disponibilidade de bateria, além de estarem potencialmente expostos a diversos usuários maliciosos. O desafio, então, surge ao observar-se que a solução para a última deficiência entra diretamente em conflito com as primeiras.

Os protocolos utilizados para comunicação segura, sejam eles aqueles utilizados por browsers (como o TLS), por terminais (como o SSH) ou até mesmo em aplicações da internet das coisas (como (**??**)), dependem de duas grandes classes de algoritmos criptográficos: os simétricos e os assimétricos. Embora as mensagens sejam cifradas com algoritmos simétricos, como o AES, o esbelecimento de um canal seguro é, de modo geral, iniciado com o uso da classe assimétrica, a qual permite a definição de um chave secreta conhecida apenas pelos participantes da conversa. Essa chave passa a ser utilizada pelo resto da sessão segura em algoritmos simétricos.

De modo geral, observa-se que algoritmos simétricos possuem implementações extremamente simples, podendo ser otimizados para diferentes configurações de processadores. É o caso do *Rijndael* (**??**), vencedor da competição que elegeu o *Advanced Encryption Standard* (AES), cuja implementação pode ser feita executando-se operações matemáticas ou com o uso de *look-up tables* que variam de tamanho: 256 bytes, 1KB ou 4KB. Desse modo, permite-se a execução do AES em processadores com diferentes configurações de memória. Tamanha flexibilidade não ocorre por acaso: o algoritmo foi desenvolvido com esse objetivo, visto que a competição incluiu critérios como requerimento de memória, eficiência computacional e simplicidade para avaliação dos candidatos (**??**).

Por outro lado, os algoritmos assimétricos possuem implementações complexas pela natureza do problema que propõem: muitos deles estão associados ao uso de conjecturas da teoria dos números sob domínios de ordem muito grande, isto é, trabalham com conjuntos cuja cardinaliade precisa de centenas ou até milhares de bits para ser representada. Um dos algoritmos mais famosos dessa classe, o RSA, é baseado na dificuldade da fatoração de inteiros e trabalha com números na ordem de 1024 bits. Pode-se perceber,

então, um dos possíveis inconvenientes que esse algoritmo trás: a aritmética de números desse tamanho não é trivial de ser implementada, principalmente se considerarmos que processadores utlizados em, por exemplo, redes de sensores trabalham com registradores de 8 ou 16 bits[1].

Nesse cenário, as técnicas baseadas em curvas elípticas (ECC) mostraram-se muito interessantes por propiciarem um mesmo nível de seguraça para um menor tamanho de chave. De acordo com (**??**), a equivalência entre os tamanhos de chaves para as duas técnicas pode ser resumida na tabela **??**. Pode-se notar que a diferença chega próximo de uma ordem de magnitude em muitos casos e, de fato, este é um dos argumentos utilizados pela RFC que adiciona algoritmos de ECC ao protocolo TLS (**??**).

ECC	RSA
163	1024
233	2048
283	3072
409	7680

Tabela 1: Tamanho de chave (em bits) necessário para que os algoritmos ECC e RSA forneçam segurança equivalente (**??**).

Apesar dos algoritmos de ECC possibilitarem uma redução no tamanho das chaves, a implementação eficiente de criptografia de chaves públicas em ambientes com recursos limitados necessitou de muitos esforços da academia, conforme relatado por (**??**). Tamanho investimento é justificado pelo crescente número de aplicações que executam em tais ambientes, como *smart cards*, redes de sensores sem fio ou etiquetas RFID. Para o caso de redes de sensores com processadores de 8 bits, apenas em 2004 demonstrou-se a viabilidade do uso de algoritmos de ECC em tais plataformas (**??**). De modo similar, a primeira implementação em hardware com consumo energético baixo o suficiente para ser alimentado passivamente foi demonstrada em 2008 (**??**). Atualmente, diversos dispositivos médicos, como marcapasos, são configurados através de um canal sem fio e possuem longevidade de 5 a 15 anos. Nesse cenário, algoritmos de curvas elípticas são empregados por proverem a seguraça e eficiência necessárias (**??**).

[1] É o caso do *Atmel ATmega* (8 bits) ou do *Texas Instruments MSP430* (16 bits)

1.1 MOTIVAÇÃO E TRABALHO PROPOSTO

Não é incomum em aplicações embarcadas que uma parcela considerável da computação seja efetuada por circuitos especializados, devido às demandas de performance e eficiência. De fato, segundo (**??**), um ASIC (*application-specific integrated circuit*) pode atingir uma eficiência de $5pJ/op$ na tecnologia CMOS de 90-nm. Por outro lado, processadores embarcados altamente eficientes atingem apenas $250pJ/op$, uma perda de 50x. É evidente, então, a troca da flexibilidade de um processador de propósito geral pela eficiência propiciada por ASICs.

Nota-se, porém, que o tempo de design de um ASIC pode ser considerável e que muitas aplicações não se adequam a inflexibilidade propiciada por essa solução. Soluções flexíveis são necessárias em classes de problemas onde o desenvolvimento de novos algoritmos ou o surgimento repentino de uma demanda inesperada são comuns. Em criptografia, um exemplo de tal demanda pode ser a descoberta de um novo ataque a algum cifrador, o que poderia requerer o aumento de algum parâmetro, como tamanho de chave, ou até mesmo a troca por outro algoritmo, o que é facilitado em implementações via software. Pode-se citar um caso onde uma especificação de hardware não acompanhou a evolução de algoritmos criptográficos: em 2011, o conjunto de instruções ARMv8 para processadores ARM foi anunciado, trazendo instruções específicas para os algoritmos SHA1 e SHA2. O primeiro foi considerado inseguro e seu uso descontinuado a partir de 2010, enquanto o segundo já teve seu sucessor escolhido via competição em 2012 (**??**).

A busca por soluções eficientes em software para criptografia, então, não deve ser desconsiderada. Em se tratanto de eficiência energética em um processador embarcado, os fatores mais impactantes são ilustrados pela figura **??**. Observa-se que o fornecimento de dados e instruções totaliza 70% da energia total consumida, tornando-se alvos importantes para otimizações. Com base nisso, o presente trabalho irá analisar o comportamento do sistema de memórias de um processador embarcado executando algoritmos de curvas elípticas, variando-se parâmetros do processador (como tamanho e associatividade das caches), parâmetros do algoritmo (como as curvas utilizadas e o tamanho de chaves) e aplicando-se técnicas de otimização voltadas à eficiência energética. Assim, será possível relacionar a influência de tais parâmetros no consumo de energia pelo sistema de memória.

O trabalho está dividido da seguinte maneira: o capítulo **??** define os conceitos básicos em criptografia e situa o contexto em que algoritmos de ECC são necessários, o capítulo **??** detalha os trabalhos correlatos, o capítulo **??** expõe os experimentos realizados, a metodologia adotada e resultados obtidos. Por fim, o capítulo **??** conclui o trabalho e levanta possibi-

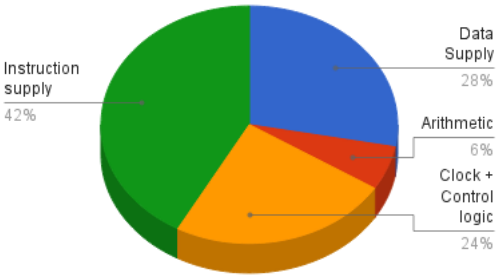

Figura 1: Consumo de energia em um processador embarcado (**??**)

lidades de trabalhos futuros.

2 CONCEITOS BÁSICOS EM CRIPTOGRAFIA

A presente seção pretende introduzir o leitor aos conceitos básicos de criptografia, iniciando por definições clássicas, seguido dos conceitos relacionados à criptografia simétrica. O material desta seção é baseado principalmente no trabalho de Goldreich (**??**) e nas notas de aula do professor Charles Rackoff (**??**), apresentando apenas uma síntese do conteúdo. O leitor interessado na bela teoria que suporta a criptografia é incentivado a consultar as referências indicadas. Embora não seja fundamental ao entendimento do resto deste trabalho, a primeira parte do capítulo é importante por situar o leitor no contexto em que a criptografia assimétrica é necessária.

Em seguida, o conceito de curva elíptica é apresentado, assim como operações necessárias à definição do logaritmo discreto. Um conhecimento básico de teoria de grupos é assumido, embora os conceitos mais importantes sejam revisados. Muitos dos exemplos e definições utilizados são baseados no anexo A do padrão IEEE 1363, (**??**).

2.1 SESSÕES SEGURAS

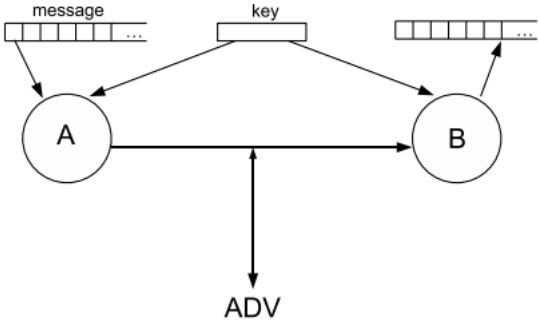

Figura 2: Modelo de uma sessão segura (**??**)

A aplicação mais tradicional de criptografia, aquela descrita por um usuário comum de informática, encontra-se no estabelecimento de **sessões seguras** (*secure sessions*). Isto é, duas pessoas, *A* e *B*, tendo um segredo em comum (daqui em diante chamado de **chave**), gostariam de se comunicar através de um canal potencialmente inseguro. Com alguma frequência, *A*

gostaria de mandar algo[1] (**texto em claro**, *plaintext*) para B; denotar-se-á por **mensagem** tudo aquilo que A gostaria de enviar para B, embora A esteja restrito a enviar uma fração de tal mensagem por vez. Considera-se também a existência de um adversário *ADV* computacionalmente irrestrito, limitado a ouvir tudo que A envia pelo canal. Informalmente, A gostaria de cifrar a mensagem de modo que *ADV* não possa descobrir nenhuma informação sobre a mesma.

Tal cenário é representado pela figura **??** e motiva a primeira definição aqui apresentada:

Definição 1 *Diremos que uma chave é um* $K \in \{0,1\}^n$*, onde* $K = K_1K_2\ldots K_n$*, e uma mensagem é um* $M \in \{0,1\}^m$*, onde* $M = M_1M_2\ldots M_m$*.*

Uma encryption function *é uma função*

$$Enc : \{0,1\}^m \times \{0,1\}^n \to \{0,1\}^*$$

Uma decryption function *é uma função*

$$Dec : \{0,1\}^* \times \{0,1\}^n \to \{0,1\}^m$$

Requer-se do par (Enc, Dec) *a seguinte* ***condição de corretude****:*

$$\forall M \in \{0,1\}^m, \forall K \in \{0,1\}^n Dec(Enc(M,K),K) = M$$

Informalmente, $Enc(M,K)$ é entendido como a mensagem M cifrada que A envia a B, o qual aplica Dec para retornar ao texto em claro. É importante insistir na condição de corretude, uma vez que ela garante que qualquer mensagem cifrada por Enc será corretamente produzida por Dec, desde que a mesma chave seja usada em ambas as funções[2].

Pode-se agora descrever o que seria uma função segura. Existem diversas definições provadamente equivalentes na literatura, por simplicidade a definição seguinte é adotada neste trabalho:

Definição 2 *O par* (Enc, Dec) *é dito* ***perfeitamente seguro*** *se, para toda* $f : \{0,1\}^* \to \{0,1\}^m$*, a seguinte proposição é válida:*

Considere o experimento em que M *é escolhida de modo aleatório de* $\{0,1\}^m$ *e* K *é escolhida de modo aleatório de* $\{0,1\}^n$*. Então deve ser verdade*

[1] Tratamos de comunicação unidirecional aqui por simplicidade, todos os conceitos podem ser expandidos para um canal bidirecional.

[2] É comum nos referimos ao par (Enc, Dec) quando queremos discutir alguma propriedade criptográfica, visto que as duas funções geralmente estão relacionadas para conferir corretude. É um exercício interessante pensar como seria fácil obter segurança sem o requerimento de corretude.

que:

$$P[f(Enc(M,K)) = M] = 1/2^m$$

Isto é, um adversário vendo a mensagem cifrada não consegue adivinhar o valor de M com probabilidade maior do que se ele simplesmente escolhesse uma mensagem aleatória. Note que o fato do adversário estar ilimitado computacionalmente é representado pelo fato de que ele é modelado como uma função arbitrária na definição.

No caso em que $n > m$, não é difícil provar a existência de um par (Enc, Dec) perfeitamente seguro: as funções $Enc(M,K) = M_1 \oplus K_1 M_2 \oplus K_2 \ldots M_m \oplus K_m$ e $Dec(X,K) = X_1 \oplus K_1 X_2 \oplus K_2 \ldots X_m \oplus K_m$ satisfazem a definição[3]. Na prática, porém, essa definição não é muito interessante, visto que gostaríamos de trabalhar com chaves pequenas (algumas centenas de bits) e mensagens de tamanho arbitrário[4]. Essa necessidade entra diretamente em conflito com o seguinte teorema:

Teorema 1 *Se $m > n$ então nenhum (Enc, Dec) é perfeitamente seguro.*

A prova, embora simples, é omitida por estar fora do escopo do trabalho e pode ser consultada nas referências indicadas. Para contornar esse teorema poderoso, adicionaremos a restrição de que *ADV* deve ser um algoritmo o qual executa em tempo polinomial (em relação a n) e definiremos uma classe de funções com a qual trabalharemos.

Definição 3 *Um **gerador de funções** (*Function Generator*) F associa a cada $n \in \mathbb{N}$ e a cada $k \in \{0,1\}^n$ uma função $F_k : \{0,1\}^n \to \{0,1\}^n$ de modo que exista um algoritmo com tempo de execução polinomial (em n) o qual compute $F_k(x)$.*

Exige-se que tais geradores de função sejam **pseudo-aleatórios**, isto é, um adversário não consegue distinguir entre um F_k (para um k escolhido aleatoriamente) e uma função escolhida aleatoriamente $f : \{0,1\}^n \to \{0,1\}^n$ com probabilidade superior a $1/n^c$ para todo c e para n suficientemente grande.

Assume-se que os algoritmos *AES* e *DES* se comportem como geradores de funções pseudo-aleatórios[5]. O primeiro, por exemplo, associa uma chave k de 128 (196 ou 256) bits com a função AES_k e é esta função que A tipicamente usa para enviar sua mensagem para B. De fato, esses algoritmos são construídos de modo as que funções geradas sejam também permutações,

[3] Tal função é conhecida como *one-time pad* na literatura.

[4] Embora o termo arbitrário seja utilizado, na prática espera-se que esse número seja muito inferior a 2^n

[5] Um pequeno abuso é feito nessa afirmação, visto que esses algoritmos são definidos para alguns poucos valores de n.

permitindo que se utilize $Enc = AES_k$ e $Dec = AES_k^{-1}$. A razão pela qual simplesmente *assumimos* que geradores de função pseudo-aleatórios existam é devido ao fato dessa ser uma afirmação mais forte que $P \neq NP$, e somos incapazes de provar isso. De fato, uma condição necessária para a segurança de toda criptografia moderna é que $P \neq NP$, embora essa não seja uma condição suficiente.[6]

2.2 CURVAS ELÍPTICAS

Até esse momento, assumiu-se que A e B haviam previamente combinado uma chave para efetuar a comunição segura. Tal hipótese pode ser suficiente para ambientes em que poucos pares precisam se comunicar, porém não é adequada para a maioria dos outros cenários. Essa observação motiva o principal objetivo da criptografia assimétrica: como fazer com que duas pessoas possam trocar alguns poucos parâmetros através de um canal inseguro e, ao final da comunicação, ambas compartilhem uma mesma chave desconhecida a qualquer adversário manipulando o canal?

Atualmente, esse objetivo é atingido com o uso de algumas conjecturas da teoria dos números. Uma delas é a dificuldade (de alguma versão) do problema do logaritmo discreto. Neste trabalho, estamos interessados na variação de curvas elípticas do problema, o assunto da presente seção.

Definição 4 *Um conjunto F com um elemento 0 e duas operações binárias $+$ e $\cdot$ é um* ***corpo finito*** *(*finite field*, ou* Galois field*) se as seguintes condições forem verdadeiras:*

1. *F é um conjunto finito*
2. *$+$ é uma operação associativa*
3. *$+$ é uma operação comutativa*
4. *0 é o elemento neutro de $+$*
5. *Cada elemento de F possui um inverso para $+$ em F*
6. *$\cdot$ é uma operação associativa*
7. *$\cdot$ é uma operação comutativa*

[6] O leitor intrigado por essa observação pode notar que não basta ser difícil quebrar um sistema criptográfico no pior caso, e é essa noção que é capturada pelo problema $P \neq NP$. É possível construir um sistema criptográfico para o qual o problema de distiguir Enc é NP-Completo, embora exista um algoritmo eficiente que o resolva em 99% dos casos (**??**).

8. *Distributividade de $\cdot$ em relação a $+$ (à esquerda e à direita)*
9. *Existência de um elemento neutro (1) para $\cdot$*
10. *Todo elemento de F diferente de 0 possui uma inversa para $\cdot$ em F*

Definição 5 *Dado um número primo p, o conjunto dos números inteiros módulo p com as operações habituais de soma e multiplicação é um corpo finito, denotado $GF(p) = \{0, 1, \ldots, p-1\}$.*

De modo similar, para um primo q e um inteiro positivo n, pode-se definir um corpo finito $GF(q^m)$ com o uso de polinomiais, embora sua definição precisa seja omitida por estar fora do escopo deste trabalho.

Definição 6 *Uma curva elíptica E sobre o corpo finito $GF(p)$, onde p é um número primo ímpar, é o conjunto:*

$$\begin{aligned} E = \{(x,y) \quad | \quad & y^2 = x^3 + ax + b \\ & \wedge \quad x, y, a, b \in GF(p) \\ & \wedge \quad 4a^3 + 27b^2 \neq 0 \ (mod\ p)\} \cup \{O\} \end{aligned}$$

onde a e b são constantes relacionadas a curva e O é chamado de ponto no infinito.

Pode-se também criar uma curva elíptica sobre $GF(2^m)$, cuja definição é dada a seguir, embora os demais conceitos desse capítulo sejam tratados apenas para o caso de $GF(p)$ para evitar demasiadas repetições. É importante lembrar, porém, que a escolha do corpo tem implicações tanto no modo como seus elementos são representados em memória como na definição de operações e algoritmos.

Definição 7 *Uma curva elíptica E sobre o corpo finito $GF(2^m)$, é o conjunto:*

$$\begin{aligned} E = \{(x,y) \quad | \quad & y^2 + xy = x^3 + ax^2 + b \\ & \wedge \quad x, y, a, b \in GF(2^m) \quad \wedge \quad b \neq 0\} \cup \{O\} \end{aligned}$$

onde a e b são constantes relacionadas a curva e O é chamado de ponto no infinito.

Definição 8 *O número de elementos de uma curva elíptica E é chamado de **ordem** de E e é denotado por $\#E(GF(q))$.*

Para ilustrar as definições, considere a curva E dada por

$$y^2 = x^3 + x + 5$$

sobre $GF(13)$. Os pontos de E são:

$$\{O,(1,4),(1,9),(3,6),(3,7),(8,5),(8,8),(10,0),(11,4),(11,9)\}$$

e $\#E(GF(13)) = 10$.

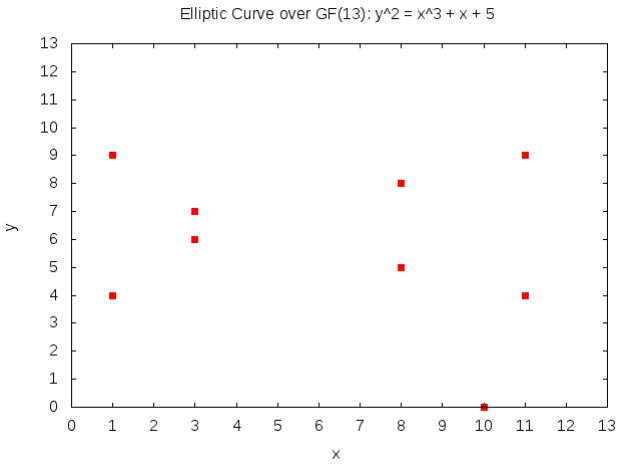

Figura 3: Exemplo de uma curva elíptica sobre $GF(13)$

A figura **??** ilustra os pontos da curva sobre $GF(13)$ e, para comparação, a mesma curva é desenhada sobre os reais na figura **??**. Como se pode observar, a troca de domínio muda consideravelmente a aparência da curva.

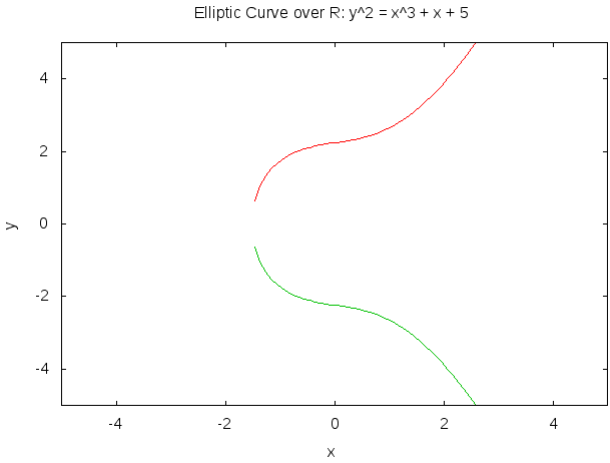

Figura 4: Exemplo de uma curva elíptica sobre $\mathbb{R}$

SOMA(P_1, P_2)

if $P_1 = O$
    output P_2 and stop.
if $P_2 = O$
    output P_1 and stop.
Set $x_1 = P_1.x$
Set $y_1 = P_1.y$
Set $x_2 = P_2.x$
Set $y_2 = P_2.y$
if $x_1 \neq x_2$
    Set $\lambda = (y_1 - y_2)/(x_1 - x_2)$
else
    if $y_1 \neq y_2$ or $y_2 = 0$
        Output O and stop.
    Set $\lambda = (3{x_2}^2 + a)/(2y_2)$
Set $x_3 = \lambda^2 - x_1 - x_2 \mod p$
Set $y_3 = (x_2 - x_3)\lambda - y_2 \mod p$
output (x_3, y_3)

Uma operação de soma $(+)$ pode ser definida sobre uma curva elíptica de modo a torná-la um grupo e, assim, poderemos definir o problema do logaritmo discreto sobre esse conjunto. Intuitivamente, tem-se que a soma de

dois pontos $P_1 + P_2$ é o ponto P_3 com a propriedade que P_1, P_2 e $-P_3$ são colineares. Para um ponto $P = (x,y)$, define-se $-P = (x,-y)$. Por clareza, definimos a soma de dois elementos de E para o caso de $GF(p)$ através do algoritmo SOMA, conforme (**??**). As linhas **??**– **??** tratam o ponto no infinito, O, como o elementro neutro da operação. No caso em que os pontos não possuam a mesma coordenada x, as linhas **??**– **??** calculam o coeficiente angular da reta secante que passa por P_1 e P_2, enquanto calcula-se o coeficiente da reta tangente a curva em um dos pontos quando $P_1 = P_2$ (linha **??**). Caso os pontos difiram apenas na coordenada y, então temos $P_1 = -P_2$ e o algoritmo retorna o elemento neutro (linhas **??**– **??**). Por fim, as linhas **??**– **??** utilizam o coeficiente mencionado anteriormente para calcular as coordenadas de P_3.

Note que, embora os termos como coefienciente angular e reta tangente sejam utilizados, os quais estão normalmente relacionados a funções sobre o domínio dos reais, o algoritmo opera sobre $GF(13)$, logo todas as operações de soma, subtração, multiplicação ou divisão que aparecem no algoritmo SOMA devem ser realizadas *mod* p. Isso ilustra uma das primeiras dificuldades de implementação de algoritmos relacionados a curvas elípticas: as operações de multiplicação e inversão em $GF(p)$ não são triviais.

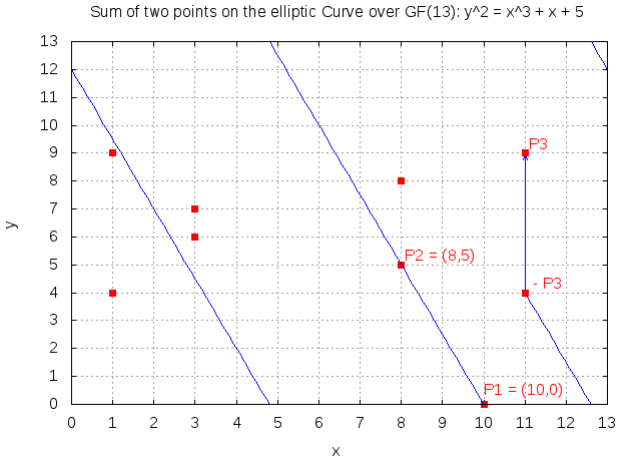

Figura 5: Somando pontos em uma curva elíptica sobre $GF(13)$

Com isto em mente, a intuição dada pelo domínio real ajuda a visualizar a soma em $GF(p)$. Utilizando a mesma curva do exemplo anterior, a figura **??** ilustra a adição dos pontos $P_1 = (10,0)$ e $P_2 = (8,5)$. Primeiro, a reta que sai de P_1 e passa por P_2 é traçada. Quando esta reta atinge os limites

do gráfico, isto é, uma de suas coordenadas atinge o valor 13 (ou 0), pode-se entender que a operação de módulo é aplicada, efetivamente continuando a reta no lado oposto do gráfico. Tal procedimento é repetido até que um ponto de E, aqui chamado de $-P_3$, seja encontrado, de modo que P_1, P_2 e $-P_3$ sejam colineares. A soma de P_1 e P_2 é dada então por P_3. Note ainda que, caso dois pontos tenham a mesma coordenada x, a reta que passa por ambos está na vertical e, portanto, nunca tocará um terceiro ponto de E que não seja o ponto no infinito, O.

Algebricamente, teríamos que:

$$\begin{aligned}\lambda &= (y_1 - y_2)/(x_1 - x_2) \quad mod\ p \\ &= (0-5)/(10-8) \quad mod\ 13 \\ &= -5/2 \quad mod\ 13 \\ &= 8*7 \quad mod\ 13 \\ &= 4\end{aligned}$$

$$\begin{aligned}x_3 &= \lambda^2 - x_1 - x_2 \quad mod\ p \\ &= 4^2 - 10 - 8 \quad mod\ 13 \\ &= 11\end{aligned}$$

$$\begin{aligned}y_3 &= (x_2 - x_3)\lambda - y_2 \quad mod\ p \\ &= (8-11)4 - 5 \quad mod\ 13 \\ &= 9\end{aligned}$$

O que é equivalente às coordenadas de P_3 obtidas geometricamente.

Como consequência da operação de adição, pode-se definir a multiplicação de um ponto por um escalar. Se $k \in \mathbb{Z}$ e $P \in E$, então:

$$kP = \begin{cases} \text{O} & : k = 0 \\ \text{(-k)(-P)} & : k < 0 \\ \text{P + (k-1)P} & : k > 0 \end{cases}$$

Definição 9 *A **ordem** de um ponto P em uma curva elíptica E é o menor inteiro positivo r tal que*

$$rP = O$$

Da álgebra, tem-se que a ordem de um ponto sempre existe e divide a ordem da curva, $\#E(GF(p))$. Além disso, se k e l são inteiros, então $kP = lP$ se, e somente se, $k \equiv l \mod p$.

Pode-se agora definir o logaritmo discreto de um ponto em uma curva elíptica:

Definição 10 *Suponha que um ponto G em uma curva E tenha ordem r, onde r^2 não divida a ordem da curva $\#E(GF(p))$. Então um ponto P satisfaz $P = lG$ para algum l se, e somente se, $rP = O$. O coeficiente l é chamado de* ***logaritmo discreto*** *de P em relação ao ponto base G.*

Uma comparação com o logaritmo tradicional pode ser feita para o entendimento da definição: enquanto $log_b x$ responde a pergunta de quantas vezes b deve ser **multiplicado** por ele mesmo até que se encontre x, o logaritmo discreto de P trabalha com quantas vezes precisa-se **somar** G até que se encontre P. Mais formalmente, tem-se (**??**):

Definição 11 *Considere uma curva elíptica E e um ponto $P \in E$ de ordem n. Dado um ponto $Q \in \langle P \rangle = \{0, P, 2P, \ldots (n-1)P\}$, o* ***problema do logaritmo discreto sobre curvas elípticas*** *(ECDLP) consiste em computar $0 \leq l \leq n-1$ tal que $Q = lP$.*

O algoritmo ingênuo para resolver uma instância de ECDLP consiste em computar os múltiplos de P até que se encontre Q, o que pode levar até n adições, impraticável para valores muito altos de n. Atualmente, os melhores algoritmos conhecidos são baseados em variações do método *Pollard's* ρ e executam em tempo $O(\sqrt{n})$ (**??**)(**??**).

2.3 *ELLIPTIC CURVE DIFFIE HELLMAN*

Em 1976, Whitfield Diffie e Martin Hellman publicaram um método para a troca de chaves em um canal inseguro (**??**) o qual é baseado no problema do logaritmo discreto de inteiros em $GF(p)$. Embora não forneça autenticação (isto é, A não possui garantias de que está de fato conversando com B[7]), ele é a base de protocolos autenticados. Pouco tempo depois, em 1978, a dificuldade da fatoração de números inteiros foi utilizada por R.L. Rivest, A. Shamir, e L. Adleman para atingir o mesmo propósito, no algoritmo

[7] Nenhum dos protocolos apresentados aqui possuirá essa característica, visto que é algo difícil de ser alcançado sem uma Infraestrutura de Chaves Públicas (*Public Key Infrastructure*) previamente estabelecida. Atualmente, isto é feito através de certificados e autoridades certificadoras. Ainda assim, tais protocolos são realistas e são utilizados em versões autenticadas dos mesmos.

conhecido como *RSA* (**??**). Em 1985, dois autores propuseram independentemente o uso de curvas elípticas para a troca de chaves ((**??**), (**??**)) com tais métodos se popularizando a partir de 2004[8].

Descreve-se aqui um dos protocolos padronizados em (**??**), lá referenciado por DL/ECKAS-DH1, ou *Discrete Logarithm and Elliptic Curve Key Agreement Scheme, Diffie-Hellman version*. *A* e *B* devem executar os seguintes passos para o estabelecimento de uma chave:

1. Em comum acordo, estabelecer um conjunto de parâmetros do domínio a ser trabalhado: os coeficientes a e b da curva, o número primo p e um ponto G de ordem r.

2. Escolher um número inteiro aleatório s no intervalo $[1, r-1]$ e computar o ponto $W = sG$. O par (s, W) consiste da chave pública e privada, respectivamente.

3. Obter da outra parte a chave pública W'.

4. Computar $P = sW' = (x_P, y_P)$. x_P será considerado o segredo compartilhado.

5. Aplicar uma função de derivação de chave sobre x_P.

As etapas, embora simples, merecem diversas considerações adicionais. Os parâmetros de domínio não podem ser gerados de forma totalmente aleatória: diversas curvas anômalas precisam ser evitadas por permitirem a simplificação do ECDLP, assim como impõe-se restrições sobre a fatoração de r para evitar o mesmo problema. Tipicamente, utilizam-se parâmetros pré-estabelecidos por padrões, como as curvas padronizadas pelo NIST (*National Institute of Standards and Technology*) (**??**)[9].

Ao receber de *B* o valor W', *A* pode utilizar o critério da definição **??** para verificar que W' é, de fato, um múltiplo de G. Nota-se também que, até esse momento, todas as mensagens foram trocadas em um canal inseguro: qualquer adversário pode ter acesso aos parâmetros de domínio, bem como a W e W' (o primeiro é enviado por *A* para *B* e o segundo por *B* para *A*). Porém apenas *A* e *B* conseguem computar facilmente o valor $P = sW' = s'W = ss'G$,

[8] Mais recentemente, as pesquisas em algoritmos baseados em Retículos (*Lattices*) ressurgiram, visto que essa é uma classe de algoritmos resistentes a ataques por computadores quânticos, ao contrário daqueles baseados em Diffie-Hellman e RSA.

[9] Essas curvas foram escolhidas, em teoria, por possuírem características adequadas para o ECDLP e por permitirem implementações eficientes das operações de grupo. Existem, no entanto, algoritmos para gerar parâmetros seguros caso o uso de curvas estabelecidas por organizações seja um inconveniente.

assumindo que o problema ECDLP seja, de fato, difícil. Por fim, aplica-se alguma função sobre o valor x_P para gerar uma string k de tamanho apropriado, a qual será utilizada em funções como o AES_k.

3 TRABALHOS CORRELATOS

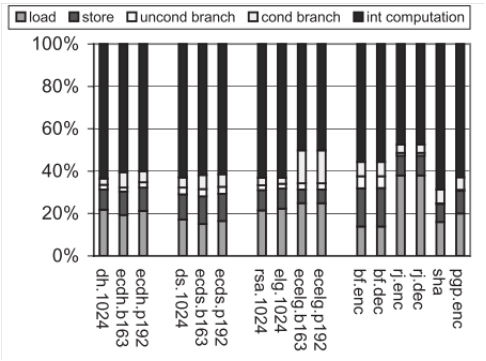

Figura 6: Distribuição dos tipos de instruções executadas durante algoritmos de ECC (segunda e terceira colunas) (**??**)

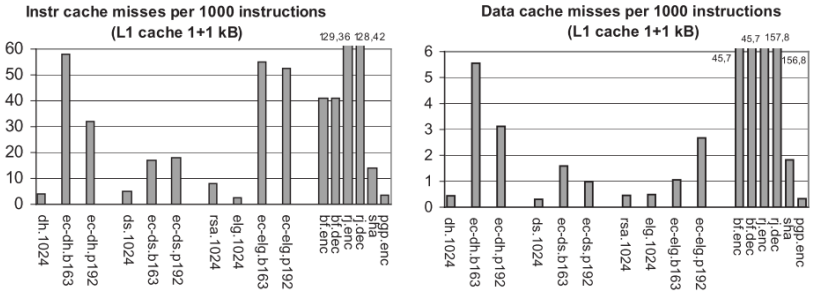

Figura 7: Comparação entre os acertos em cache dos algoritmos de ECC (colunas ec-dh) e AES (colunas bf) (**??**)

As primeiras publicações as quais investigaram a performance de sistemas de criptografia baseados em curvas elípticas possuiam um grande enfoque em tempo de execução. Em (**??**), realizou-se um levantamento extensivo de diversas implementações para as operações aritméticas em $GF(2^m)$, assim como algoritmos para multiplicação por escalar em curvas, e o tempo de execução em um processador de desktop da época foi reportado. (**??**) apresentaram uma biblioteca para um processador ARM de 32 bits com operações

para curvas em $GF(p)$, reportando tempo de execução para o ECDSA (*Elliptic Curve Digistal Signature Algorithm*), o qual utiliza as mesmas primitivas que ECDH.

O trabalho de (**??**), a partir do simulador Superscalar para a arquitetura ARM, foi o primeiro a reportar dados do comportamento do sistema de memória para algoritmos de ECC, embora para uma única configuração de cache. Além disso, encontram-se dados sobre CPI e tempo de execução por procedimento do algoritmo, bem como a proposta de uma instrução para multiplicação de polinomiais e o impacto que isso teria no algoritmo. Tal proposta é interessante visto que, anos depois, o conjunto de instruções ARMv8 trouxe uma instrução similar, embora não se encontrou nenhuma técnica utilizando ela na prática.

Em uma continuação desse trabalho, (**??**) realizaran uma análise da distribuição dos tipos de instrução executadas pelo algoritmo, onde pode-se observar que instruções de ALU totalizam 60% do total de executadas (figura **??**), resultado confirmado pelos trabalhos de (**??**) e (**??**). Outra contribuição consiste na comparação do *miss rate* entre algoritmos simétricos e assimétricos, onde pode-se observar que os algoritmos de ECC são muito menos intensivos na cache de dados quando comparados ao AES, embora detalhes da implementação deste não sejam reportados. Além disso, a diferença entre o *miss rate* na cache de dados e na de instruções chega a uma ordem de magnitude, conforme a figura **??**.

Outro conjunto de trabalhos passa a focar em processadores utilizados em sensores, como (**??**), o qual comparou o tempo de execução, uso de memória e tamanho do código para os algoritmos RSA e ECC em um ATmega128. As conclusões apresentadas em tal trabalho indicam que os algoritmos baseados em RSA podem ser até uma ordem de magnitude mais lentos, embora espere-se que essa diferença seja menor para processadores com tamanho de palavra maior. Uma implementação detalhada para plataforma de sensores MICA, a qual utiliza o processador ATmega128, é descrita em (**??**), embora não sejam informados dados de eficiência enérgica. Um trabalho por (**??**) compara esquemas de ECC com esquemas baseados em identidades e pareamentos, onde conclui-se que os primeiros possuem um desempenho superior. Uma segunda contribuição do trabalho consiste em demonstrar a importância do uso apropriado de otimizações voltadas a arquitetura alvo, visto que os autores foram os primeiros a considerar a instrução de *multiply and accumulate* do processador MSP430 ao desenvolverem sua implementação. Nessa mesma plataforma, (**??**) reporta o consumo energético ao variar-se a frequência operada bem como a presença de multiplicador em hardware, atributo o qual não está presente em todas as versões do MSP430.

Quatro anos após a prova em silício de que algoritmos de ECC podem

ser utilizados em RFID (**??**), apresentou-se uma implementação em software para a plataforma RFID conhecida como WISP (**??**). Neste trabalho, apenas a menor curva padronizada pelo NIST é utilizada, reportando tempo de execução de 1.6 segundos para multiplicação de um ponto da curva por um escalar.

O trabalho de (**??**) avalia o uso de curvas elípticas com o recém lançado processador Cortex M0+ da ARM (32 bits), propondo um novo algoritmo para multiplicação em $GF(q)$. Além disso, o custo enérgico de cada instrução executada pelo processador é avaliado e o desempenho dos algoritmos criptográficos calculado com base em tais valores. Infelizmente, o trabalho se concentra em curvas especiais e não naquelas padronizadas. De modo similar, (**??**) explora duas formas especiais de curvas elípticas (*Montgomery* e *Twisted Edwards*) bem como a equivalência existente entre elas para diminuir o custo das operações de grupo. A importância do uso de curvas padronizadas é discutida em (**??**), onde os autores desenvolvem clones VHDL (precisos em ciclo) dos processadores ATmega128, MSP430 e Cortex M0+ e simulam o uso de tais curvas. Destaca-se que o processador ARM alcança a melhor eficiência energética e tempo de execução enquanto o MSP430 ocupa a menor área em silício e demanda a menor potência. Por fim, deve-se mencionar que o estado da arte em termos de velocidade encontra-se no trabalho publicado por (**??**), o qual, embora não utilize curvas padronizadas, afirma trabalhar sobre um corpo finito mais compatível com outras implementações de ECC.