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

Um agradecimento

“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 evalute 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

Introducao!

2 CONCEITOS BÁSICOS EM CRIPTOGRAFIA

A presente seção pretende introduzir o leitor aos conceitos básicos de criptografia, iniciando pelas definições clássicas, seguido dos conceitos relacionados à criptografia simétrica. O material desta seção é baseado principalmente no trabalho de Goldreich (GOLDREICH, 2006) e nas notas de aula do professor Charles Rackoff (RACKOFF, 2014), 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, a definição de curva elíptica é apresentada, 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, (IEEE..., 2000).

2.1 SESSÕES SEGURAS

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 motiva a primeira definição aqui apresentada:

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

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

Uma encryption function *é uma função*

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

Uma decryption function *é uma função*

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

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].

Podemos 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* $f : \{0,1\}^* \rightarrow \{0,1\}^m$*, 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 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 arbitrária. 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 $Enc(M,K) = M_1 \oplus K_1 M_2 \oplus K_2 ... M_m \oplus K_m$ e $Dec(X,K) = X_1 \oplus K_1 X_2 \oplus K_2 ... X_m \oplus K_m$

[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.

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 clareza. 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 polynomial (em n) o qual compute $F_k(x)$.*

Nós iremos exigir que tais geradores de função sejam **pseudo-aleatórios**, isto é, um adversário não consegue distinguir entre um F_k 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, permitindo que se utilize $Enc = AES_k$ e $Dec = AES_k^{-1}$. A razão pela qual simplesmente *assumimos* que geradores de função pseudo-aleatórios existem é devido ao fato dessa ser uma afirmação mais forte que $P \neq NP$, e somos incapazes de provar isso.

2.2 CURVAS ELÍPTICAS

Até esse momento, assumimos que A e B haviam previamente combinado uma chave para efetuar a comunição segura. Tal hipótese

[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.

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 possuam 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 *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, porén, que a escolha do grupo tem implicações tanto em como seus elementos são representados em memória como na definição de algumas operações e algoritmos.

Definição 5 *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 6 *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 $\#E(GF(13)) = 10$.

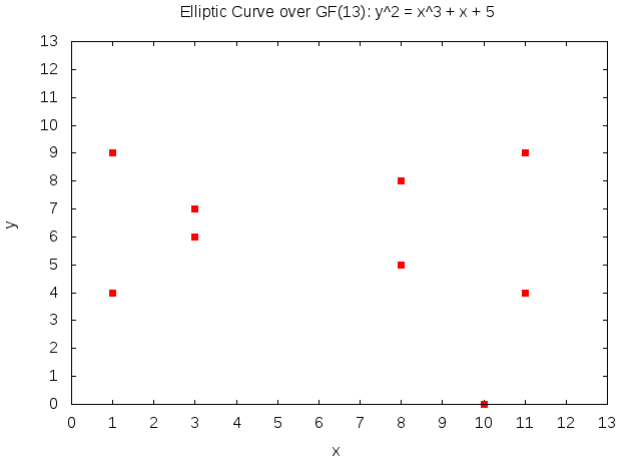

Figura 1: Exemplo de uma curva elíptica sobre $GF(13)$

A figura 1 ilustra os pontos da curva sobre $GF(13)$ e, para comparação, a mesma curva é desenhada sobre os reais na figura 2. Como se pode observar, a troca de domínio muda consideravelmente a aparência da curva.

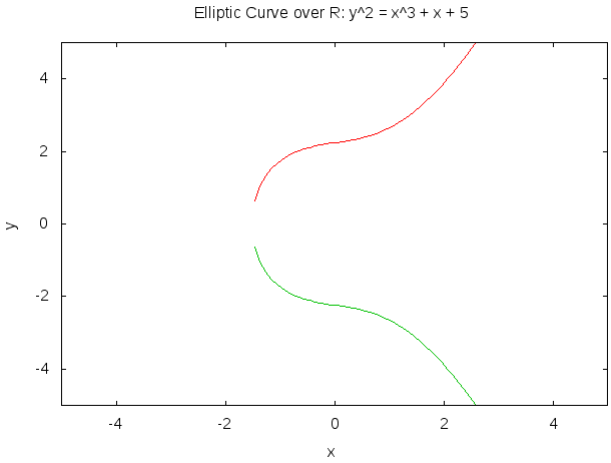

Figura 2: Exemplo de uma curva elíptica sobre $\mathbb{R}$

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 (IEEE..., 2000). As linhas 1–4 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 9–10 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 14). Caso os pontos difiram apenas na coordenada y, então temos $P_1 = -P_2$ e o algoritmo retorna o elemento neutro (linhas 12–13). Por fim, as linhas 15–16 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.

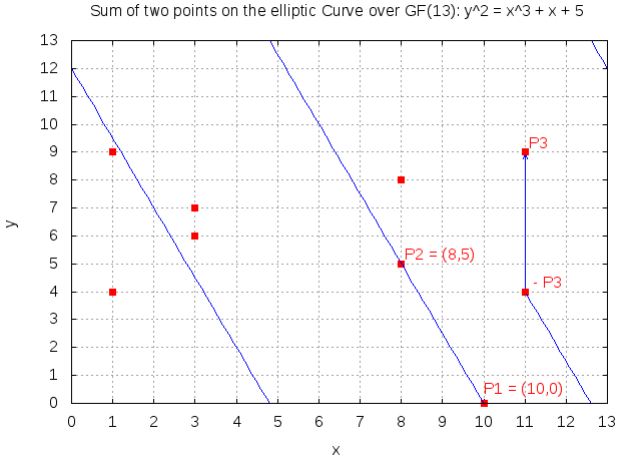

Figura 3: Somando pontos em uma curva elíptica sobre $GF(13)$

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 3 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 do 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:

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

Definição 7 *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 8 *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.

2.3

REFERÊNCIAS

GOLDREICH, O. **Foundations of Cryptography: Volume 1**. New York, NY, USA: Cambridge University Press, 2006. ISBN 0521035368.

IEEE Standard Specifications for Public-Key Cryptography. [S.l.], Aug 2000. 1-228 p.

RACKOFF, C. **Fundamentals of Cryptography**. 2014. University Lecture. Disponível em: <http://www.cs.toronto.edu/~rackoff/2426f14/>.