# Biblioteca Criptográfica baseada em Traços XTR

Diogo Haruki Kykuta

Trabalho de conclusão de curso

Orientador: Prof. Dr. Paulo Barreto

São Paulo, Dezembro de 2012

# Agradecimentos

Em certa parte do curso, um certo professor introduziu o conceito de *ursinho*. Quando travamos em um projeto, seja por bug ou por não ter ideia de como prosseguir, devemos explicar nosso problema em voz alta para alguém, mesmo que essa pessoa não entenda nada do seu problema. Muitas vezes, ao organizar os pensamentos para conseguir explicá-lo em voz alta, encontramos uma solução. Essa pessoa é o *ursinho*.

Gostaria de agradecer a todos os *ursinhos* que me ajudaram na realização desse trabalho.

# Resumo

Em criptografia, existem algumas maneiras de garantir a segurança dos dados. Dentre elas, existem aquelas baseadas em teoria dos números. Dessas, destacam-se as baseadas em fatoração em números primos e as baseadas no problema do logaritmo discreto.

Naquelas baseadas no problema do logaritmo discreto, a ideia é escolher um subgrupo multiplicativo de um corpo no qual o logaritmo seja uma operação intratável. Nas implementações atuais, existe um uso relativamente grande de curvas elípticas, por sua notória redução no tamanho das chaves, mas com isso vem uma dificuldade muito grande em encontrar uma boa curva e realizar as operações sobre esse corpo.

XTR é uma maneira eficiente de representar os elementos de um corpo através de seus traços, que são pertencentes a um subcorpo menor, sem precisar representar elementos do corpo base. Dessa forma, gastamos menos espaço representando cada elemento, mantendo fácil a escolha dos parâmetros e as operações não tão complicadas.

Nesse trabalho, será feita uma implementação de uma biblioteca que implemente XTR, permitindo as operações necessárias com os traços para que consiga, de fato, substituir o próprio elemento, e geração de parâmetros que obedeçam as propriedades desejadas. Além disso, dois protocolos foram implementados com XTR, para que sua viabilidade fosse testada.

# Sumário

# Parte Objetiva

# Capítulo 1

# Introdução

## 1.1 Motivações e Objetivos

Muitas são as vezes em que queremos proteger os dados, mas nem sempre isso é considerado algo viável, por insuficiência de recursos computacionais. Para um certo nível de segurança, se tentarmos obter chaves pequenas, acabamos tendo operações muito complicadas, exigindo mais processamento e tendo uma implementação bem mais complexa. Por outro lado, ao buscar operações simples, em geral somos obrigados a usar chaves muito longas para alcançar o nível de segurança estabelecido.

Em [LV00], Lenstra e Verheul apresentaram uma maneira de representar eficientemente elementos de um corpo inteiro a partir de seu traço, pertencente a um subcorpo. Ao evitar a representação de elementos do corpo base, e tendo uma redução de $\frac{1}{3}$ no número de bits para representar cada elemento, esse método pode ser um meio termo: chaves não muito extensas e operações não muito complexas.

E assim, espera-se que esse método permita uma biblioteca simples e compacta, que não exija um tamanho absurdo de chaves, dessa forma sendo capaz de ser implantada em ambientes com recursos limitados, como por exemplo sistemas embarcados, e ter um processo de criação de chaves muito mais simples que outros protocolos.

# Capítulo 2

# Base Teórica

Nessa seção, serão apresentadas alguns conceitos necessários para a compreensão do XTR. As definições e resultados a seguir são definidos em [Fra89] e nesse trabalho não serão exibidas as provas dos mesmos.

## 2.1 Operações binárias

**Definição 1** **Operação binária**
Uma *operação binária* sobre um conjunto $S$ é uma regra que atribui a cada par ordenado de elementos de $S$ algum elemento de $S$.

□

**Definição 2** **Comutatividade**
Uma operação binária $*$ sobre um conjunto $S$ é dita *comutativa* se e somente se $a * b = b * a$ para todo $a, b \in S$.

□

**Definição 3** **Associatividade**
Uma operação binária $*$ sobre um conjunto $S$ é dita *associativa* se e somente se $(a*b)*c = a*(b*c)$ para todo $a, b, c \in S$.

□

**Definição 4** **Fechamento**
Seja $*$ uma operação binária sobre um conjunto $A$. Seja $B$ um subconjunto de $A$. Dizemos que essa operação é *fechada* em $B$ se e somente se $a * b \in B$ para todo $a, b \in B$.

□

## 2.2 Grupos

**Definição 5 Grupo**
Um *grupo* $(G, *)$ é um conjunto $G$ juntamente com uma operação binária $*$, tais que os seguintes axiomas são satisfeitos:

- A operação binária $*$ é *associativa*.
- Existe um elemento $e$ em $G$ tal que $e * x = x * e = x$ para todo $x \in G$. Esse elemento $e$ é chamado *elemento neutro* para $*$ sobre $G$.
- Para cada $a$ em $G$, existe um elemento $a'$ em $G$ que satisfaz $a * a' = a' * a = e$. O elemento $a'$ é chamado de *inverso de a em relação a* $*$.

**Notação** Um grupo passará a ser representado por $G$.

□

**Definição 6 Grupo Abeliano**
Um grupo $G$ é dito *abeliano* se a operação binária $*$ obedece a propriedade *comutativa*.

□

**Notação** Se o grupo for abeliano, a operação binária associada será representada por $+$ ao invés de $*$.
**Notação** Se a operação binária for comutativa, representaremos o *elemento inverso* de $a$ por $-a$.
**Notação** Se a operação binária puder ou não ser comutativa (representada neste trabalho por $*$), representaremos o *elemento inverso* de $a$ por $a^{-1}$.

**Definição 7 Grupo Finito**
Um grupo $G$ é dito *finito* se o conjunto de elementos de $G$ for finito.

□

**Definição 8 Ordem de um Grupo**
Se $G$ é um grupo finito, então a ordem de $G$, representado por $|G|$, é o número de elementos em $G$.

□

## 2.3 Subgrupos

**Definição 9 Subgrupo**
Se $H$ é um subconjunto de um grupo $G$ tal que a operação binária associada a $G$ é fechada em $H$, então $H$ é um grupo sob essa operação binária. Dizemos então que $H$ é um *subgrupo* de $G$.

□

**Definição 10** **Subgrupo cíclico**
Seja $G$ um grupo e $a \in G$.

$$H = \{a^n \mid n \in \mathbb{Z}\}$$

é um subgrupo de $G$ e é o menor subgrupo de $G$ que contém $a$.

$H$ é o *subgrupo cíclico* gerado por $a$.

**Notação** Um subgrupo cíclico gerado por $a$ será representado por $\langle a \rangle$.

□

## 2.4 Anéis

**Definição 11** **Anel**
Um anel $(R, +, \cdot)$ é um conjunto $R$ com duas operações binárias $+$ e $\cdot$. Onde $+$ é a adição e $\cdot$ é a multiplicação sobre $R$ e os seguintes axiomas são satisfeitos:

- $(R, +)$ é um grupo abeliano.
- A multiplicação $\cdot$ é *comutativa*.
- Para todo $a, b, c \in R$, então vale a *distributiva pela esquerda*, ou seja $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$ e a *distributiva pela direita*, ou seja $(a + b) \cdot c = (a \cdot c) + (b \cdot c)$.

□

**Definição 12** **Anel Comutativo**
Um anel em que a multiplicação é *comutativa* é um *anel comutativo*.

□

**Definição 13** **Anel com Unidade**
Um anel $R$ com elemento neutro em relação à multiplicação, ou seja, existe $1 \in R$ tal que $1 \cdot x = x \cdot 1 = x$, é dito um *anel com unidade*.

□

**Definição 14** **Anel de Divisão**
Seja $R$ um anel com unidade. Um elemento $u$ de $R$ é uma unidade de $R$ se ele não tem inverso multiplicativo em $R$. Se todo elemento diferente de zero de $R$ é uma unidade, então $R$ é um *anel de divisão*.

□

## 2.5 Corpos

**Definição 15** **Corpo**
Um *corpo* é um anel de divisão comutativo.

□

**Definição 16 Corpos Finitos**

Um corpo $(\mathbb{F}, +, \cdot)$ é dito *finito* se $\mathbb{F}$ for um conjunto finito.

Corpos Finitos também são conhecidos como *corpos de Galois*. Um corpo de Galois pode ser representado como $GF(p^n)$, onde $p$ é um número primo e $n$ um número natural, e $p^n$ é a *ordem* do corpo. Qualquer corpo finito tem como ordem uma potência de primo. E todo corpo de ordem prima (quando $n$ é 1) é isomorfo ao corpo $\mathbb{Z}/p\mathbb{Z}$.

□

**Definição 17 Grupo Multiplicativo de um Corpo**

Dado um corpo $(F, +, \cdot)$, o grupo $(F^*, \cdot)$ é dito o *grupo multiplicativo* desse corpo.

□

# Capítulo 3

# XTR

Nesse capítulo será apresentada a teoria por trás do XTR: uma breve explicação dos parâmetros e as propriedades a que eles devem atender, como funciona a representação de traços, nível de segurança adquirido e o ganho de desempenho em relação ao corpo base.

## 3.1 Introdução

XTR (originalmente ECSTR - *Efficient and Compact Subgroup Trace Representation*) é um método apresentado por Lenstra e Verheul em [LV00] para uma representação eficiente de traços. Ele pode ser usado junto com qualquer protocolo criptográfico baseado no uso de subgrupos, levando a uma diminuição no *overhead* computacional. Essa conjunção não leva a nenhuma perda de segurança dos protocolos originais, e uma breve explicação sobre isso será dada em 3.7. A prova completa pode ser encontrada no artigo original.

Nesse método, ao invés de representar os elementos de $GF(p^6)$, o corpo base, são representados seus traços, pertencentes a $GF(p^2)$. Como é usada uma representação por base normal, na qual cada coeficiente é um elemento de $GF(p)$, seria necessária uma sêxtupla para representar cada elemento de $GF(p^6)$ enquanto apenas uma dupla é necessária para um elemento de $GF(p^2)$.

Dessa forma, esperando-se uma segurança equivalente a chaves de 1024 bits, basta termos $p^6$ com esse tamanho. Ou seja, basta $p$ ter aproximadamente $1024/6 \approx 170$ bits. Então, cada elemento de $GF(p^2)$ utiliza cerca de 340 bits para ser representado.

Esse protocolo utiliza aritmética totalmente em $GF(p^2)$, conseguindo ainda assim manter a segurança de $GF(p^6)$, sem precisar representar elementos de $GF(p^6)$. Assim, cada operação entre elementos utiliza menos operações em $GF(p)$, o corpo a que pertencem os coeficientes da representação por base normal. Essa redução do número de operações será melhor discutida em 3.6.

## 3.2 Parâmetros

Para o XTR ser utilizado, existem dois parâmetros que devem ser definidos: $p$ e $q$. Esses dois números devem ser primos. $p$ será utilizado para definir o tamanho do corpo base e do subcorpo a que pertencem os traços, ou seja, $GF(p^6)$ e $GF(p^2)$, respectivamente. $q$ definirá a ordem do subgrupo que deve ser escolhido. Na seção 3.4 será explicado mais sobre a escolha do subgrupo.

$p$ e $q$ devem ter certas propriedades:

- $p \equiv 2 \ mod \ 3$
- $q$ divide $p^2 - p + 1$

Em 5.3 está descrito o algoritmo implementado na biblioteca para escolher esses parâmetros.

## 3.3 Traços

Durante toda a teoria do XTR, fala-se muito no traço de um elemento, então aqui será dada uma definição dessa operação. O traço é uma função de um corpo em um subcorpo. Neste trabalho, será uma função de $GF(p^6)$ em $GF(p^2)$.

Para um elemento $h \in GF(p^6)$, seus conjugados sobre $GF(p^2)$ são $h$, $h^{p^2}$ e $h^{p^4}$. Assim como em todo o restante do trabalho, a prova não será mostrada aqui, e para quem tiver interesse, pode-se verificá-la no artigo [LV00].

**Definição 18 Traço**
O *traço* sobre $GF(p^2)$ de $h \in GF(p^6)$ é a soma dos conjugados sobre $GF(p^2)$ de $h$.
**Notação** O traco sobre $GF(p^2)$ de $h$ será representado como $Tr(h)$.
$Tr(h) = h + h^{p^2} + h^{p^4}$

□

$$Tr(h)^{p^2} = (h + h^{p^2} + h^{p^4})^{p^2} = h^{p^2} + h^{p^4} + h^{p^6}$$

Como a ordem de $h$ divide $p^6 - 1$, $h^{p^6} = h$. Assim,

$$Tr(h)^{p^2} = Tr(h)$$

Portanto, $Tr(h) \in GF(p^2)$.
Se $g \in GF(p^6)$ tem ordem que divide $p^2 - p + 1$, então seus conjugados são $g$, $g^{p-1}$ e $g^{-p}$.

**Definição 19 F(c,X)**
Para $c \in GF(p^2)$, seja $F(c, X)$ o polinômio $X^3 - cX^2 + c^pX - 1 \in GF(p^2)[X]$.
Esse polinômio tem raízes $h_0$, $h_1$ e $h_2 \in GF(p^6)$, não necessariamente distintas.
Definimos $\tau(c, n) = h_0^n + h_1^n + h_2^n$, para $n \in \mathbb{Z}$.
**Notação** Será utilizada a notação $c_n$ para indicar $\tau(c, n)$.

□

No artigo, são apresentados alguns métodos que, conhecendo $c, c_{n-1}, c_n, c_{n+1}$, consegue-se calcular $c_{2n}, c_{2n-1}, c_{2n+1}$ e $c_{n+2}$.

## 3.4 Subgrupo XTR

Escolher um elemento $g$ do corpo base que tenha uma certa ordem pode não ser uma tarefa muito simples, já que elementos desse corpo nunca são representados. Mas não é necessário encontrar tal $g$, basta que seja encontrado um traço desse elemento. Formalizando, basta encontrar $Tr(g) \in GF(p^2)$ tal que $g \in GF(p^6)$ tenha ordem $q$. Esse $q$ já foi definido em 3.2.

No artigo de Lenstra e Verheul, é mostrado que se um elemento $c \in GF(p^2)$ atende a certas propriedades, uma certa modificação dele é um traço de um elemento $g$ de ordem $q$.

Seja $c$ um elemento de $GF(p^2)$. Se

- $c \notin GF(p)$
- $c_{p+1} \notin GF(p)$
- $c_{(p^2-p+1)/q} \neq 3$

então $c_{(p^2-p+1)/q} = Tr(g)$ para algum $g$ com ordem $q$.

A implementação desse algoritmo na biblioteca é apresentada em 5.4

## 3.5 Chaves XTR

Como muitos protocolos utilizam as mesmas informações em sua chave pública, é de se esperar que possa ser feita uma implementação única e padrão de chave para ser usada em qualquer versão XTR dos protocolos. Isso foi feito e alguns detalhes sobre informações presentes nas chaves são mostrados a seguir:

Como informação privada, tem-se a chave $k \in \mathbb{Z}$, que é um número aleatório entre 1 e $q-2$.

As informações públicas são:

- $p$, parte dos parâmetros obrigatórios (veja 3.2)
- $q$, também parte dos parâmetros XTR.
- $Tr(g)$, onde $g$ é o gerador do subgrupo XTR, de ordem $q$.
- $Tr(g^{k-1}), Tr(g^k), Tr(g^{k+1})$, onde $k$ é a chave privada.

## 3.6 Custo das operações

A análise feita por Lenstra e Verheul do custo esperado de multiplicações para realizar certas computações é apresentada a seguir, e os detalhes de como se fazem essas operações gastando tal número de multiplicações (sobre $GF(p^2)$) é apresentado na seção 5.2.

Os custos apresentados na tabela abaixo são custos médios esperados das operações em número de multiplicações em $GF(p)$. Alguns dos valores da tabela são mostrados por Lenstra e Verheul em seu artigo, outros foram calculados (grosseiramente) e é mostrada a seguir uma pequena explicação do número de operações.

| | $GF(p^2)$ | $GF(p^6)$ |
|---|---|---|
| $x^p$ | 0 | ... |
| $x^2$ | 2 | 14.4 |
| $x * y$ | 3 | 18 |
| $x * y - y * z^p$ | 4 | ... |
| $x^a$ | $5 * \log_2(a)$ | $23.4 * \log_2(a)$ |
| $x^a * y^b$ | $5 * \log_2(a) + 5 * \log_2(b) + 3$ | $27.9 * \log_2(\max(a, b))$ |

- Para $x^a$, com $x \in GF(p^2)$ e $a \in \mathbb{Z}$:

  Varrendo os bits do expoente $a$, o valor é elevado ao quadrado por bit, ao custo de 2 multiplicações cada. Caso o bit seja 1, esse valor é multiplicado por $x$ ao custo de 3 multiplicações. Então, para cada bit, no máximo são realizadas 5 multiplicações. Logo, $5 * \log_2(a)$ multiplicações no pior caso.

- Para $x^a * y^b$, com $x, y \in GF(p^2)$ e $a, b \in \mathbb{Z}$:

  Usando o argumento de cima, é fácil encontrar o valor $5*\log_2(a)+5*\log_2(b)+3$ multiplicações para cada.

Vê-se claramente, dessa forma, que realizar operações sobre $GF(p^2)$ é muito menos custoso do que realizar operações sobre $GF(p^6)$. XTR é o primeiro sistema a utilizar operações sobre $GF(p^2)$ mantendo a segurança de $GF(p^6)$, de acordo com [LV00].

## 3.7 Segurança

A segurança dos protocolos que se baseiam no Problema do Logarítmo Discreto, ao usar XTR, devem ser revistas, já que essas versões desses protocolos irão atuar sobre outros subgrupos. Para isso, deve ser definida a nova versão do problema a ser estudado: XTR-DL (*XTR-Discrete Logarithm*), que é o DL sobre o subgrupo XTR.

Seja $\langle\gamma\rangle$ um grupo multiplicativo de ordem prima $\omega$. Alguns dos possíveis ataques são atacar o grupo multiplicativo, usando uma variação do Crivo de Corpos Numéricos, ou atacar o subgrupo, que tomaria $O(\sqrt{\omega})$ operações em $\langle\gamma\rangle$, usando por exemplo um método baseado no paradoxo do aniversário.

Ou seja, a dificuldade do problema DL em $\langle\gamma\rangle$ depende do tamanho do corpo envolvente mínimo de $\langle\gamma\rangle$ e depende também da sua ordem prima $\omega$. Se $GF(p^t)$ é o corpo envolvente mínimo de $\langle\gamma\rangle$ e $\omega$ é grande o suficiente, então o problema do logaritmo discreto em $\langle\gamma\rangle$ é tão difícil quanto o problema DL em $GF(p^t)$.

Os parametros usados no XTR são escolhidos de forma que o corpo envolvente mínimo do grupo XTR seja igual a $GF(p^6)$, com $p$ e $q$ grandes o suficiente para adquirir a segurança desejada.

Foi mostrado que o problema XTR-DL é (1,1)-equivalente ao problema DL em $\langle g\rangle$. A prova pode ser encontrada no artigo original. Assim, utilizar XTR não proporciona mudança na segurança dos protocolos originais.

# Capítulo 4

# Protocolos usando XTR

Nesta seção, serão mostrados alguns protocolos que podem ser alterados para a utilização do XTR. Serão apresentados um algoritmo de criptografia (seção 4.1) e um de assinatura (seção 4.2). Também serão mencionados alguns outros protocolos em que a alteração é possível, sem apresentar uma implementação.

## 4.1 Criptografia XTR-ElGamal

A criptografia ElGamal, adaptada para o XTR, passa-se da seguinte forma:

A chave utilizada é a chave XTR, definida na biblioteca. Ela contém as informações sobre $p$, $q$, $Tr(g)$ e $Tr(g^k)$ públicas, que serão necessárias para criptografar uma mensagem; e $k$ privado para decriptografar. Existem algumas outras informações na chave pública que não serão usadas nesse protocolo ($Tr(g^{k-1})$ e $Tr(g^{k+1})$), mas já foi dito que essas informações extras não prejudicam a segurança dos protocolos e de suas versões XTR.

Para criptografar uma mensagem, basta seguir o algoritmo abaixo:

ElGamal-Encrypt($M, publica$)
1 $params \leftarrow publica.parametros$
2 $Tr(g) \leftarrow publica\,.Tr(g)$
3 $Tr(g^k) \leftarrow publica\,.Tr(g^k)$
4 $service \leftarrow$ **new** XTRService($params$)
5 $x \leftarrow$ Random$(1, q-2)$
6 $a \leftarrow$ service.Exponenciacao-Simples$(Tr(g), x)$
7 $simetrica \leftarrow$ service.Exponenciacao-Simples$(Tr(g^k), x)$
8 $enc \leftarrow$ Criptografia-Simetrica-Aux$(M, simetrica)$
9 **return**$(a, enc)$

Para a Criptografia-Simetrica-Aux pode ser utilizado qualquer protocolo simétrico como auxiliar, desde que respeitem a interface `CriptografiaSimetrica` definida no projeto XTR-ElGamal.

Apesar de ter sido implementada uma criptografia simétrica bem simples, qualquer protocolo simétrico pode ser utilizado. Uma boa alternativa seria utilizar o *Keccak* para isso, devido ao tamanho variável das chaves e mensagens, para evitar a necessidade do tratamento do tamanho das chaves e da *chain* para criptografar entradas maiores do que o protocolo espera. *Keccak* [BDPA11] é uma função de hash, mas com certas alterações propostas a funções esponja [BDPVA12], que é o caso do *Keccak*, podemos duplexá-la e criar um protocolo capaz de criptografar e decriptografar a partir de chaves simétricas.

## 4.2 Assinatura Schnorr

A assinatura Schnorr pode ser alterada para utilizar XTR facilmente. Basta utilizar os traços de $g$ quando deveria ser utilizado o próprio $g$ no algoritmo tradicional.

### 4.2.1 Assinando uma mensagem

*$M$ = mensagem (em vetor de bytes)*

SCHNORR-SIGN($M$, *privada*)
1 *params* $\leftarrow$ *privada.parametros*
2 $Tr(g) \leftarrow$ *privada* $.Tr(g)$
3 *service* $\leftarrow$ **new** XTRSERVICE(*params*)
4 $r \leftarrow$ *privada* $.Tr(g^k)$
5 $e \leftarrow$ HASH(CONCATENATE($M, r$))
6 $s \leftarrow k - x * e$
7 **return**$(s, e)$

### 4.2.2 Verificando uma assinatura

$(s, e)$ assinatura da mensagem $M$

SCHNORR-VERIFY($M$, *publica*, $s, e$)
1 *params* $\leftarrow$ *publica.parametros*
2 $r_v \leftarrow$ *service.Exponenciacao* $-$ *Simples*$(Tr(g), s)$
3 $e_v \leftarrow$ HASH(CONCATENATE($M, r_v$))
4 **if** $e_v = e$
5 **then return** Verdadeiro
6 **else return** Falso

## 4.3 Alguns outros protocolos

Vários outros protocolos também podem ser adaptados para utilizar XTR. Basta que ele utilize subgrupos multiplicativos de ordem conhecida como base de sua segurança. Como exemplos, pode-se citar Diffie-Hellman e Assinatura Cega.

Apesar dos exemplos implementados (4.1 e 4.2) terem sido simples, seu diferencial para os outros protocolos não se dá pela variedade de operações, mas sim pela quantidade. Em outras palavras, os demais protocolos podem ser implementados usando uma quantidade diferente (quanto mais complexo, espera-se que uma quantidade maior) de exponenciações simples e duplas. Assim, mesmo com a simplicidade dos protocolos implementados, eles cobrem utilizações desses dois tipos de exponenciação, o que deve bastar para garantir a funcionalidade da biblioteca.

# Capítulo 5

# Implementação

## 5.1 Representações usadas

Mostrou-se necessária uma classe para representar um elemento do subcorpo ($GF(p^2)$), chamada de `XTRElem`. Ela é composta basicamente de um vetor de duas posições, armazenando os valores de $x_1$ e $x_2$ da representação matemática $x = x_1\alpha + x_2\alpha^2$, $x_1$ e $x_2$ são elementos de $GF(p)$. O vetor utilizado para armazená-los é um vetor de `BigInteger`.

O construtor da classe `XTRElem` precisa de mais informações, mas como será mencionado algumas vezes, então uma versão simplificada está definida abaixo, para facilitar. No projeto, ele requer algumas informações sobre o $p$ ou sobre o corpo inteiro ao qual seus coeficientes pertencem. Aqui, isso será deixado de lado por praticidade.

XTRE$\textsc{lem}(x_1, x_2)$

Os coeficientes desse novo elemento são $x_1 \mod p$ e $x_2 \mod p$, respectivamente.

## 5.2 Operações sobre GF($\mathbf{p^2}$)

Para as seguintes subseções, sejam $x = x_1\alpha + x_2\alpha^2$ e $y = y_1\alpha + y_2\alpha^2$. No projeto, suas representações são `XTRElem` com coeficientes $\{x_1, x_2\}$ e $\{y_1, y_2\}$, respectivamente.

### 5.2.1 Adição

XTR-A$\textsc{dd}(x, y)$
**return** XTRE$\textsc{lem}(x_1 + y_1, x_2 + y_2)$

### 5.2.2 Multiplicação

XTR-$\textsc{mul}(x, y)$
$temp \leftarrow x_1 * y_2 + x_2 * y_1$
**return** XTRE$\textsc{lem}(x_2 * y_2 - temp, x_1 * y_1 - temp)$

### 5.2.3 Exponenciação a $p$

XTR-$\textsc{exp}$-P$(x)$
**return** XTRE$\textsc{lem}(x_2, x_1)$

### 5.2.4 Multiplicação por um escalar

XTR-SCALAR-MUL($x, scalar$)
  **return** XTRELEM($scalar * x_1, scalar * x_2$)

## 5.3 Escolha dos parâmetros

Lenstra e Verheul apresentam duas maneiras de escolher parâmetros em [LV00]. Apenas uma delas foi implementada nessa biblioteca e ela é descrita a seguir:

Sejam $P$ e $Q$ os tamanhos (em número de bits) de $p$ e $q$ a serem gerados, respectivamente. Temos, na biblioteca, os valores padrão de $P = 170$ e $Q = 160$, mas podem ser passados como parâmetro.

GENERATE-PARAMETERS($P, Q$)
1 $p \leftarrow 0$
2 $q \leftarrow 0$
3 **while** IS-NOT-PRIME($p$)
4  **do** $r \leftarrow$ RANDOM$(0, 2^{Q/2})$
5  $q \leftarrow r^2 - r + 1$
6 **while** IS-NOT-PRIME($q$)
7  **do** $k \leftarrow$ RANDOM$(0, 2^{P-Q})$
8  $p \leftarrow r + kq$
9 **return** $p, q$

## 5.4 Seleção do subgrupo

Um método para encontrar $Tr(g)$ com $g$ atendendo as propriedades descritas anteriormente é essencial, visto que $g \in GF(p^6)$ nunca é representado. O algoritmo implementado para que tal $Tr(g)$ seja encontrado é descrito a seguir, em pseudocódigo.

GENERATE-TRACE-ORDER-Q($q$)
1 $c \leftarrow$ RANDOM-XTRELEM()
2 **if** $c \in GF(p)$
3  **then goto** 1
4 $c_{p+1} \leftarrow$ EXPONENCIACAO-SIMPLES$(c, p + 1)$
5 **if** $c_{p+1} \in GF(p)$
6  **then goto** 1
7 $c_{p^2-p+1} \leftarrow$ EXPONENCIACAO-SIMPLES$(c, p^2 - p + 1)$
8 **if** $c_{p^2-p+1} = 3$
9  **then goto** 1
10 **return** $c_{p^2-p+1}$

Esse $c_{p^2-p+1}$ é traço de um elemento que tenha ordem $q$, ou seja, $c_{p^2-p+1} = Tr(g)$, com $|g| = q$.

## 5.5 Organização do projeto

O trabalho está dividido em 3 projetos diferentes: XTR-lib, XTR-Elgamal e XTR-Schnorr, para facilidade de compreensão e utilização. XTR-Elgamal e XTR-Schnorr são implementações de protocolos de criptografia e de assinatura, respectivamente, com as modificações necessárias para usarem XTR. Esses projetos dependem de XTR-lib.

### 5.5.1 XTR-lib

Esse projeto contém os códigos referentes a biblioteca XTR: definição de elementos dos grupos usados, criação de chaves e parâmetros, serviços para a execução de operações usando tais grupos.

Algumas das classes importantes estarão descritas brevemente aqui.

#### GaloisField

Representa um corpo de Galois. Deve ser passado para o construtor um `BigInteger` $p$.
Aqui estão definidas as operações sobre $GF(p)$.

#### XTRElem

Representa um elemento do subcorpo XTR ($GF(p^2)$).
Seu construtor espera 2 `BigInteger` e um `GaloisField` para criar uma instância.

#### XTRTrincaElem

Quando está sendo feita a exponenciação dupla, o cenário é um pouco diferente. Outras informações são necessárias, tais como $Tr(g^{k-1})$ e $Tr(g^k + 1)$. Por esse motivo, temos $h$ definida como uma `XTRTrincaElem` na chave pública, contendo $Tr(g^{k-1})$, $Tr(g^k)$ e $Tr(g^{k+1})$.

#### XTRParameters

Classe que representa os parâmetros necessários para o XTR, ou seja: $p$, para definir $GF(p)$, $GF(p^2)$ e $GF(p^6)$; $q$, a ordem do subgrupo, com a propriedade de $q$ dividir $p^2 - p + 1$. Essa classe guarda uma referência para o Corpo de Galois $GF(p)$.

#### XTRKey

Representação de uma chave XTR. Ela é composta basicamente de uma chave pública e uma chave privada. Nenhuma das informações é armazenada diretamente na `XTRKey`, o que faz com que tenhamos redundância de dados que apareçam tanto na chave privada quanto na chave pública.

#### XTRPrivateKey

Representação da chave privada.
Uma chave privada contém as seguintes informações:

| | |
|---|---|
| $p$ | Inteiro que determina o corpo $GF(p^6)$ |
| $q$ | Ordem do subgrupo em que vamos trabalhar, atendendo as propriedades definidas em 3.2 |
| $Tr(g)$ | Elemento de $GF(p^2)$, com $g \in GF(p^6)$ e ordem de $g = q$ |
| $k$ | Um inteiro secreto que garante a segurança do protocolo, onde $1 < k < q - 2$ |

#### XTRPublicKey

Representação da chave pública.
Uma chave pública contém as seguintes informações:

| | |
|---|---|
| $p$ | Inteiro que determina o corpo $GF(p^6)$ |
| $q$ | Ordem do subgrupo em que vamos trabalhar, atendendo as propriedades definidas em 3.2 |
| $Tr(g)$ | Elemento de $GF(p^2)$, com $g \in GF(p^6)$ e ordem de $g = q$ |
| $Tr(g^k)$ | Elemento de $GF(p^2)$, traço de $g \in GF(p^6)$ elevado a $k$, inteiro secreto definido acima, na chave privada. |

**Exceptions**

Para a organização do projeto e detecção mais fácil de erros, foram criadas algumas *Exceptions* para alguns casos. Elas são listadas abaixo.

| | |
|---|---|
| `GaloisFieldArithmeticException` | Algum erro ocorrido durante a execução de alguma operação em $GF(p)$ |
| `XTRInvalidParametersException` | Lançada quando é detectado que os parâmetros XTR utilizados não são válidos (não atendem as propriedades especificadas). |
| `XTRNullParametersException` | Exceção que deve aparecer quando algum valor null é passado quando se espera um XTRParameter em construtores ou funções. |
| `XTRInvalidKeyException` | Lançada no momento em que se detecta que a chave não é válida. Isso pode acontecer quando a chave privada não condiz com a chave pública ou algum valor interno obrigatório da chave é nulo. |
| `XTRNullKeyException` | null foi passado como parâmetro para alguma função que esperava uma XTRKey. |

**Serviço de Operações XTR**

Esse serviço provê os seguintes métodos:

- `generateKeys`
  Gera um par de chaves XTR. Devolve a `XTRKey` criada.
- `exponenciacaoSimples`
  Recebe um $Tr(g)$ e um inteiro $n$ e calcula $Tr(g^n)$
- `exponenciacaoDupla`
  Recebe $Tr(g)$, $Tr(g^k)$ para um $k$ desconhecido, $a, b$ inteiros e calcula $Tr(g^a * g^{kb})$
- `Sn`
  Dado um elemento $c \in GF(p^2)$ e um $n$ inteiro, calcula a trinca $(c_{n-1}, c_n, c_{n+1})$. Ela é utilizada para calcular a exponenciação simples.

Na implementação padrão, um XTRParameter deve ser passado no construtor, e a geração de chaves se faz utilizando esses parâmetros e o método random1AtehQ devolve um elemento $a \in GF(p)$, $1 < a < q - 2$.

### 5.5.2 XTR-Elgamal

Uma implementação do protocolo Elgamal de criptografia, assistida de um protocolo de criptografia simétrica.

### 5.5.3 XTR-Schnorr

Uma implementação da assinatura Schnorr, alterada para funcionar em cima dos traços, utilizando a XTR-lib.

# Capítulo 6

# Resultados

Ao final do projeto, tivemos como produto uma biblioteca em JAVA que implementa as ideias do XTR, permitindo as operações necessárias para que qualquer versão XTR dos protocolos criptográficos possam ser implementados; e dois protótipos de protocolos criptográficos em suas versões XTR.

O trabalho foi inicialmente feito seguindo o paradigma da Orientação de Objetos, o que fez com que essa biblioteca não fosse tão enxuta quanto poderia ser. Durante essa fase, a biblioteca utilizava cerca de 33.6 kB, contando os arquivos *.class* gerados.

Após uma fase de tentar enxugar a biblioteca, essa passou a ocupar 19.4 kB, usando o mesmo critério. Mas para isso, foi perdida a implementação padrão de chave XTR que havia sido feita, assim como todas as verificações de validade das propriedades de parâmetros e chaves. Todas as `Exceptions` do projeto foram removidas. Apesar disso, a biblioteca ainda é utilizável para a realização das operações necessárias.

Existe ainda muito o que enxugar na biblioteca, podendo torná-la ainda mais compacta, mas isso teria como custo uma menor legibilidade do código, o que foi considerado como não atrativo para esse trabalho.

# Parte Subjetiva

# Capítulo 7

# Desafios e Frustrações

Ao iniciar o projeto, não tinha ideia do que esperar. Ainda não tinha feito a matéria de Criptografia (estaria cursando-a no primeiro semestre da execução desse trabalho) mas tinha muito interesse. Interesse suficiente para decidir fazer meu trabalho de formatura nessa área. Como o professor Routo não estava mais aceitando alunos para orientação de TCC, segui uma dica de um amigo e fui procurar o professor Paulo Barreto, na Poli, para me orientar. Ele sugeriu que fizemos esse trabalho. Passei os primeiros meses me dedicando a entender os conceitos básicos de criptografia e relembrar o que tinha aprendido em Álgebra 2, matéria que foi muito importante na execução do projeto.

Ao iniciar a implementação do trabalho, deparei-me com diversos erros: varredura de bits sendo feita errada, problemas na representação dos elementos, encontrar uma representação em um vetor de bytes para os traços... Ou até mesmo erros de digitação. O início da monografia, por outro lado, foi bem tranquilo. Peguei um modelo de dissertação do IME, fiz alguns ajustes básicos e comecei a escrever.

Mas, perto do período da entrega da versão preliminar, tive alguns problemas de saúde, que me causaram um certo tempo com o trabalho parado e um ritmo meio lento depois. Além disso, no segundo semestre, estava matriculado em MAC0430 - Algoritmos e Complexidade Computacional, mas ela exigia muito esforço, fisico e mental, tomando tempo de estudo que pretendia destinar ao TCC. Decidi, então, trancar a matéria.

Apesar desses problemas, o trabalho foi seguindo... E nas horas que desanimava, tinha uma criaturinha chata que ficava falando: "OW, VAI FAZER SEU TCC". Devo muito desse trabalho a ela.

Acredito que esse trabalho me proporcionou uma visão muito boa da área de criptografia, muito melhor do que eu jamais teria somente cursando a matéria. Vi, por exemplo, que não nasci para isso. Não me traz a satisfação que achava que teria... O gosto que achei que teria... E com isso, veio a dúvida da área do meu mestrado.

# Capítulo 8

# Relação entre o trabalho de formatura e disciplinas do BCC

**MAC0122** - `Princípios de Desenvolvimento de Algoritmos`

Essa matéria é uma das principais no curso. Ela dá uma base muito importante para o desenvolvimento de programas de uma forma elegante e organizada. Além de dar algumas noções de eficiência dos algoritmos.

**MAC0338** - `Análise de Algoritmos`

Para todo programa em que se exige eficiência, essa disciplina fornece a bagagem acadêmica necessária para a análise formal de sua eficiência. Nesse trabalho, onde se esperava uma diminuição no *overhead*, essa bagagem foi extremamente útil.

**MAC0336** - `Criptografia para Segurança de Dados`

Este é um trabalho de criptografia, e nessa matéria adquiri os conhecimentos e noções básicas para a compreensão de diversas partes do trabalho, mesmo ele sendo mais focado na parte algébrica para servir de auxiliar nos protocolos.

**MAT0138** - `Álgebra 1 para Computação`

Essa matéria introduz as ideias de *Inteiros módulo primo*, teorema de Fermat, dentre algumas outras noções importantes para a compreensão da teoria por trás desse modelo de chave pública.

**MAT0213** - `Álgebra 2`

As noções de grupos, anéis e corpos são apresentadas nessa matéria. Noções essas que foram primordiais durante o decorrer desse trabalho, como pode ser visto na parte objetiva

# Capítulo 9

# Próximos passos

Como qualquer implementação na área de criptografia, essa biblioteca precisa tornar-se estável antes de poder ser utilizada. Ou seja, precisa-se verificar se existem falhas na implementação que possam causar perdas de segurança.

Além disso, essa biblioteca foi feita em java com orientação de objetos, o que faz com que ela não fique tão pequena. Assim, pode-se deixá-la enxuta para que possa ser utilizada para os fins idealizados no começo desse trabalho. Apesar de ter sido criada uma versão compacta da biblioteca, foi feito somente o básico para a redução de seu tamanho. Ainda pode-se fazer algumas otimizações podendo custar a perda do entendimento do código.

Existe um outro artigo [SL01] que apresenta uma maneira de otimizar certas operações feitas pela biblioteca, mas pode ter como custo uma biblioteca maior. Um estudo do ganho de desempenho e do custo no espaço necessário para armazenar a biblioteca mostra-se um próximo passo interessante.

# Referências


[BDPA11] G. Bertoni, J. Daemen, M. Peeters e G. Van Assche. The keccak reference. Submission to NIST (Round 3), 2011. 13

[BDPVA12] Guido Bertoni, Joan Daemen, Michaël Peeters e Gilles Van Assche. Duplexing the sponge: single-pass authenticated encryption and other applications. Em *Proceedings of the 18th international conference on Selected Areas in Cryptography*, SAC'11, páginas 320–337, Berlin, Heidelberg, 2012. Springer-Verlag. 13

[Fra89] J.B. Fraleigh. *A first course in abstract algebra*. Addison-Wesley Pub. Co., 1989. 5

[LV00] Arjen K. Lenstra e Eric R. Verheul. The xtr public key system. Em *CRYPTO'00*, páginas 1–19, 2000. 3, 9, 10, 11, 16

[SL01] Martijn Stam e Arjen K. Lenstra. Speeding up xtr. Em *ASIACRYPT'01*, páginas 125–143, 2001. 27