

**EDUARDO DE SOUZA CRUZ  
GEOVANDRO CARLOS CREPALDI FIRMINO PEREIRA  
RODRIGO RODRIGUES DA SILVA**

## **APLICATIVO SEGURO DE SMS**

Texto apresentado à Escola Politécnica da Universidade de São Paulo como requisito para a conclusão do curso de graduação em Engenharia de Computação, junto ao Departamento de Engenharia de Computação e Sistemas Digitais (PCS).

São Paulo  
2008

**EDUARDO DE SOUZA CRUZ  
GEOVANDRO CARLOS CREPALDI FIRMINO PEREIRA  
RODRIGO RODRIGUES DA SILVA**

**APLICATIVO SEGURO DE SMS**

Texto apresentado à Escola Politécnica da Universidade de São Paulo como requisito para a conclusão do curso de graduação em Engenharia de Computação, junto ao Departamento de Engenharia de Computação e Sistemas Digitais (PCS).

Área de Concentração:

Engenharia da Computação

Orientador:

Prof. Dr. Paulo Sérgio Licciardi Messeder Barreto

# **AGRADECIMENTOS**

Agradeço a... etc etc etc

## **RESUMO**

Este trabalho consiste na especificação e implementação de um sistema que garanta serviços de segurança na troca de mensagens SMS entre aparelhos de telefonia celular. Ao longo da monografia, apresentaremos aspectos da nossa solução, um esquema criptográfico inovador que viabilizou a implementação, métricas, resultados de testes de desempenho, além de considerações sobre o andamento do trabalho.

# **ABSTRACT**

This work consists in the specification and implementation of a system which is able to guarantee security services while exchanging SMS between

# SUMÁRIO

**Lista de Figuras**

**Lista de Tabelas**

**Lista de abreviaturas e siglas**

<b>1</b>	<b>Introdução</b>	<b>11</b>
1.1	Cenário . . . . .	12
<b>2</b>	<b>Objetivos</b>	<b>13</b>
2.1	Escopo . . . . .	13
2.2	Métricas . . . . .	14
<b>3</b>	<b>Discussão</b>	<b>17</b>
<b>4</b>	<b>Preliminares teóricas</b>	<b>19</b>
4.1	Curvas Elípticas . . . . .	19
4.2	Emparelhamentos Bilineares . . . . .	19
4.3	Criptografia baseada em identidades . . . . .	19
4.4	Alternativas existentes . . . . .	19
4.5	Revisão literatura . . . . .	19
<b>5</b>	<b>Escolha de um esquema criptográfico adequado</b>	<b>20</b>

5.1	BLMQ . . . . .	20
5.1.1	Testes de viabilidade . . . . .	20
5.2	BDCPS . . . . .	21
5.2.1	Definições . . . . .	22
5.2.2	Testes de viabilidade . . . . .	25
5.3	Conclusão . . . . .	26
<b>6</b>	<b>Especificação do sistema</b>	<b>28</b>
6.1	Requisitos funcionais . . . . .	28
6.2	Requisitos não-funcionais . . . . .	28
6.3	Arquitetura . . . . .	28
6.4	Classes . . . . .	28
6.5	Casos de uso . . . . .	28
6.6	Especificação do protocolo . . . . .	28
6.6.1	RequestMyQAMessage . . . . .	28
6.6.2	HerelsYourQAMessage . . . . .	29
6.6.3	AuthenticationMessage . . . . .	31
6.6.4	SigncryptedException . . . . .	32
6.7	Escolha de parâmetros . . . . .	33
<b>7</b>	<b>Implementação</b>	<b>34</b>
7.1	Metodologia . . . . .	34
7.2	Ambiente de desenvolvimento . . . . .	34



7.3	Bibliotecas usadas . . . . .	34
7.4	Problemas encontrados . . . . .	34
<b>8</b>	<b>Resultados</b>	<b>35</b>
8.1	Desempenho . . . . .	35
8.2	Comparação com outras soluções . . . . .	35
<b>9</b>	<b>Conclusão</b>	<b>36</b>
9.1	Análise dos resultados . . . . .	36
9.2	Possíveis desenvolvimentos futuros . . . . .	36
	<b>Referências</b>	<b>37</b>
	<b>Apêndice A - Apêndice</b>	<b>39</b>

## LISTA DE FIGURAS

1	RequestMyQAMessage . . . . .	29
2	HereIsYourQAMessage . . . . .	31
3	AuthenticationMessage . . . . .	32
4	SignCryptedMessage . . . . .	33

## LISTA DE TABELAS

1	Testes com BLMQ . . . . .	21
2	Testes com o novo esquema (chaves de 127 bits) e comparação com o RSA . . . . .	26
3	Testes com o novo esquema (chaves de 160 bits) e comparação com o RSA . . . . .	27

## LISTA DE ABREVIATURAS E SIGLAS

**SMS** - Short Message Service

# 1 INTRODUÇÃO

Atualmente não existem soluções universalmente adotadas para garantir segurança em mensagens SMS. Pelo método tradicional, as mensagens trafegam pela rede celular de forma insegura, passando obrigatoriamente por pelo menos um intermediário não 100% confiável: a operadora do serviço de telefonia. As mensagens podem ficar armazenadas em texto plano no banco de dados da operadora (NG, 2006), de forma que pessoas mal intencionadas infiltradas no sistema podem ser capazes de visualizar, alterar e até enviar mensagens em nome de outra pessoa. Há também outros métodos para interceptar mensagens SMS. (ENCK et al., 2005).

O sistema desenvolvido foi designado por "Aplicativo Seguro de SMS", ou, em uma forma abreviada e internacionalizada mais adequada ao mercado, "Secure-SMS", ou ainda "SSMS". O objetivo do sistema é prover uma camada de segurança a nível de aplicação para mensagens SMS em redes de telefonia móvel. O software é capaz de assinar, cifrar, decifrar e verificar mensagens enviadas por SMS, de forma a garantir a identidade do remetente, e garantir que ela somente poderá ser lida pelo destinatário real, oferecendo assim os serviços de segurança: autenticidade, confidencialidade e integridade.

Apesar de implementado especificamente para ser executado em um telefone celular, garantindo a comunicação pessoal sigilosa, o mesmo esquema pode ser facilmente portado para outras plataformas que têm acesso à comu-

nicação por SMS, como PDAs ou servidores. Desta forma, o esquema pode ser aplicado em áreas que requerem alto nível de segurança da informação que trafega nas redes de telefonia móvel, por exemplo aplicações militares, bancárias, e de comércio eletrônico.

## **1.1 Cenário**

O ambiente em questão não se mostra muito propício para práticas criptográficas. A largura de banda é muito pequena, visto que em cada mensagem SMS podem ser trafegados apenas 140 bytes binários. Além disto, existem limitações de processamento no dispositivo celular, que podem comprometer a usabilidade de um esquema criptográfico tradicional.

Talvez devido a estas dificuldades, o cenário atual não apresenta uma grande variedade de soluções abrangindo objetivos similares aos de nosso sistema, e não há uma solução universalmente adotada.

## **2 OBJETIVOS**

O objetivo do trabalho é quebrar o paradigma atual, desafiar as dificuldades existentes, e produzir um sistema que garante segurança no intercâmbio de mensagens SMS entre dispositivos celulares sem afetar a usabilidade do serviço. Além disto, o nível de segurança obtido deve ser equiparável ao nível de segurança de aplicações de segurança vigentes atualmente, como por exemplo uma transação bancária pela Internet pelo protocolo HTTPS.

### **2.1 Escopo**

O software a ser desenvolvido será designado por "Sistema de SMS Seguro". O objetivo do software é prover uma camada de segurança a nível de aplicação para mensagens SMS em redes de telefonia móvel. O software deverá fornecer alguns serviços básicos de segurança como confidencialidade, integridade e autenticidade das mensagens SMS, permitindo ao usuário assinar, cifrar, decifrar e verificar mensagens enviadas por SMS. As soluções adotadas no projeto envolvem criptografia de chave pública, criptografia baseada em identidade e também esquemas auto-certificados. Tal adoção deve dispensar a existência de um diretório de chaves públicas, uma vez que seria necessário o uso de certificados que ocupam muita banda do ambiente restrito utilizado e encarece o processo.

O software será aplicável em áreas que requeiram segurança da infor-

mação que trafega nas redes de telefonia móvel. Alguns exemplos são aplicações militares, bancárias, comunicação pessoal sigilosa e comércio eletrônico. Os principais benefícios do sistema são a sua flexibilidade, podendo ser adaptado às necessidades dos clientes, e leveza, já que sua arquitetura é restrita à camada de aplicação: o software opera sobre a camada de aplicação do modelo OSI, sendo transparente à arquitetura interna da rede móvel. Essas duas qualidades tornam possível sua viabilidade em diversos cenários.

O Sistema de SMS Seguro não é responsável pela confiabilidade e disponibilidade de entrega das mensagens independente de sua natureza, essa função é de responsabilidade do fornecedor da aplicação que utiliza o protocolo, no caso de mensagens SMS a responsabilidade de entrega será da operadora telefônica, no caso de serviços de e-mail a responsabilidade de entrega é conferida ao servidor que fornece o serviço, etc.

No caso do serviço SMS, se houver qualquer tipo de clonagem de telefone o Sistema Seguro de SMS não mais se responsabiliza pelo serviço de segurança, uma vez que o número do telefone é o identificador do usuário e não mais poderá garantir a autenticidade do emissor das mensagens produzidas pelo dispositivo clonado. Porém, a vantagem do sistema é que ele é capaz de perceber a clonagem quando uma nova mensagem é enviada a partir do telefone clonado.

Uma outra característica não tratada é a irretratabilidade, ou seja, o sistema não fornece este serviço, porém pode ser feita uma pesquisa de viabilidade futura para tornar possível implementar o serviço de irretratabilidade.

## **2.2 Métricas**

A seguir, definimos as métricas desejadas.



- Nível de segurança: Desejamos que o nível de segurança de nosso protocolo seja equiparável ao nível de segurança do RSA usando chaves de 1024 bits, que pode ser calculado como  $2^{1024}$ .
- Tempo de espera: Consiste nos tempos para cifrar e verificar uma mensagem. Baseando-se em aplicações já existentes e satisfazendo os requisitos de usabilidade de nosso projeto, estimamos que um intervalo de espera para processamento de uma mensagem de no máximo 5 segundos seja tolerável pelo usuário.
- Tamanho máximo de mensagens do protocolo: Consiste da soma dos bytes úteis da mensagem com os bytes de controle do algoritmo. Implementações SMS baseadas em *Sun Wireless Messaging API (WMA)* podem dividir uma única mensagem em, no máximo, 3 segmentos, totalizando 399 bytes binários (ORTIZ, 2002). Porém para evitar problemas devido à segmentação, decidimos que as mensagens do nosso protocolo deverão caber em apenas 1 segmento, o que nos dá um tamanho de 140 bytes binários para cada mensagem do protocolo.
- Tamanho das chaves privada/pública: Devido às limitações de banda, estabeleceu-se que cada o tamanho chave usada não deverá exceder 200 bits. No entanto, essa restrição não deve comprometer o nível de segurança desejado.
- Tamanho máximo de uma mensagem de texto a ser cifrada e enviada: Uma mensagem cifrada deverá caber em uma única mensagem do protocolo, cujo tamanho máximo foi definido em 140 bytes. Porém nem todos os bytes poderão ser usados para a mensagem, pois existirá um overhead do protocolo, devido à cabeçalhos e dados da assinatura. Estabelecemos então um tamanho máximo de 70 bytes para

uma mensagem de texto. Desta forma, ficam reservados outros 70 bytes para o overhead e a assinatura.

- Tamanho do certificado: Devido às limitações de banda, estabeleceu-se que o tamanho do certificado de uma chave não deverá exceder 512 bits. Desejamos poder transferir o certificado em um único SMS, sem comprometer o espaço necessário para o overhead do protocolo.<sup>1</sup>

---

<sup>1</sup>Mais tarde o leitor verá que optamos por um esquema que não usa certificados, ou seja, esta métrica será atendida com 0 bits de tamanho de certificado

### 3 DISCUSSÃO

Sabendo que um certificado digital típico ocupa entre 2KB e 4KB, nota-se aqui que uma solução baseada em infra-estrutura convencional de chaves públicas inviabilizaria completamente o sistema: antes de se enviar uma mensagem SMS segura para algum usuário, seria necessário receber o certificado desse usuário particionado em 15 a 30 mensagens SMS, além de enviar em resposta outro certificado em mais 15 a 30 mensagens SMS. Esse esforço precisaria ser efetuado novamente para cada novo destinatário a quem determinado usuário desejasse enviar mensagens, ou em cada caso de renovação ou revogação de certificado. Some-se a isto o espaço ocupado por uma única assinatura convencional, tipicamente de 128 bytes por estar baseada no algoritmo RSA com 1024 bits; este *overhead* seria duplicado com o requisito de cifrar e assinar a mensagem, isto é, tomaria 256 bytes do espaço disponível.

Por outro lado, a manutenção de um diretório confiável de chaves públicas, típico de sistemas de criptografia convencionais, seria impraticável em uma rede de telefonia celular. Uma solução tecnológica baseada em alternativas à criptografia convencional é, portanto, imprescindível.

Sendo assim, foi considerado o uso de criptografia em curvas elípticas com assinatura baseada em identidades, de acordo com o conceito proposto inicialmente por Shamir (SHAMIR, 1984). Aprofundando-se na especificação, percebeu-se ainda que a chave pública do usuário poderia ser estabelecida

essencialmente a partir de sua identificação única no sistema, ou seja, seu próprio número de celular. Desse modo, a criptografia em curvas elípticas baseada em identidades com emparelhamentos bilineares parecia ser capaz de atender aos requisitos do nosso aplicativo.

## **4 PRELIMINARES TEÓRICAS**

### **4.1 Curvas Elípticas**

### **4.2 Emparelhamentos Bilineares**

### **4.3 Criptografia baseada em identidades**

### **4.4 Alternativas existentes**

### **4.5 Revisão literatura**

## **5 ESCOLHA DE UM ESQUEMA CRIPTOGRÁFICO ADEQUADO**

### **5.1 BLMQ**

A primeira tentativa de solução adotava o esquema de cifrassinatura baseada em identidades BLMQ (BARRETO et al., 2005).

O esquema foi escolhido por, aparentemente, atender aos requisitos estabelecidos inicialmente. O BLMQ era notadamente mais eficiente que esquemas de criptografia baseada em identidades anteriores, como o de Boneh-Franklin (BONEH; FRANKLIN, 2001), o que poderia tornar o uso desse tipo de criptografia viável em ambientes produtivos. Além disso, uma assinatura de 160 bits garantiria um nível de segurança de aproximadamente 80 bits equivalente ao do RSA de 1024 bits (KALISKI, 2003).

#### **5.1.1 Testes de viabilidade**

O esquema foi implementado na linguagem de programação Java, e testes foram realizados em um aparelho celular Nokia 6275.

O desempenho observado inicialmente foi insatisfatório, não atendendo aos requisitos de usabilidade estabelecidos na especificação. Foram feitas tentativas de melhoria do desempenho, como variação do tamanho das chaves, uso de diferentes funções de emparelhamento (Ate, Eta) (FREEMAN; SCOTT; TESKE, 2006), e implementações com diferentes bibliotecas que for-

necessem a classe *BigInteger* - a implementação da Sun se mostrou mais eficiente do que a implementação da Bouncy Castle. Algumas adaptações no esquema em si foram feitas, como inversão da ordem das curvas utilizadas.

Os melhores resultados obtidos são apresentados na tabela 1.

Tabela 1: Testes com BLMQ	
Operação	Tempo (s)
Inicialização das classes	128.9
Emparelhamento Eta	4.2
Emparelhamento Ate	3.9

Como estes tempos não atendiam aos requisitos de usabilidade do projeto, fez-se necessário buscar alternativas. Estas dificuldades serviram como motivação para a criação de um esquema inovador. Como resultado de pesquisas realizadas, foi idealizado um novo esquema, apresentado em (BARRETO et al., 2008) e brevemente descrito a seguir.

## 5.2 BDCPS

Visando resolver os problemas encontrados com a utilização do BLMQ, um novo esquema foi criado, e designado por BDCPS. Um artigo (BARRETO et al., 2008) foi submetido para o SBSEG'08, sediado em Gramado - RS.

Na criação do novo esquema, em vez de utilizarmos exclusivamente criptografia e assinaturas baseadas em identidades, estendemos um esquema de criptografia sem certificados com um esquema de assinatura convencional, mas utilizando técnicas baseadas em identidades para validar a chave pública, evitando o uso de certificados (BARRETO et al., 2008). A nova técnica mescla esses dois paradigmas, garantindo baixo tempo de cifrassinatura e de verificação, tamanhos de chaves dentro dos limites adotados e níveis de segurança satisfatórios.

O esquema proposto integra esquemas preexistentes como as assinaturas BLMQ e Schnorr (SCHNORR, 1991) e o esquema isento de certificados de Zheng (ZHENG, 1997). Neste esquema, a geração das chaves dos usuários dispensa a necessidade de uma autoridade certificadora e a utilização de certificados para validar sua chave pública. Estas características implicaram em importantes, mas não únicas, melhorias em relação ao esquema anteriormente implementado e serão discutidas na seção "Análise da Proposta". Por criptografia convencional entende-se o fato de o usuário poder escolher seu par de chaves não certificado, ou seja, ele escolhe apropriadamente uma chave privada e gera sua chave pública a partir dela. Desse modo, somente o usuário gerador de seu par de chaves convencional conhece sua chave privada, e este fato elimina a possibilidade de "Key Escrow", que é o comprometimento da chave e conseqüentemente das mensagens com ela encriptadas. O par de chaves não certificado é combinado com a solução de criptografia baseada em identidades para que seja posteriormente validado por outro usuário do sistema.

### 5.2.1 Definições

O novo esquema consiste dos seguintes algoritmos:

- **Setup:** Algoritmo gerador do conjunto dos parâmetros públicos necessários. O algoritmo escolhe um parâmetro de segurança  $k$  e define:

$n$  : Um inteiro primo de  $k$  bits.

$(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$  Grupos de mapeamento bilinear de ordem  $n$

$e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ : Emparelhamento eficientemente computável e não-degradado.



$P \in \mathbb{G}_1$ ,  $Q \in \mathbb{G}_2$ : Os pontos geradores dos grupos  $\mathbb{G}_1$  e  $\mathbb{G}_2$  respectivamente.

Resumos criptográficos (*hashes*)

$$h_0 : \mathbb{G}_T^2 \times \{0, 1\}^* \rightarrow \mathbb{Z}_n^*,$$

$$h_1 : \mathbb{G}_T \times \{0, 1\}^* \rightarrow \mathbb{Z}_n^*,$$

$$h_2 : \mathbb{G}_T \rightarrow \{0, 1\}^*,$$

$$h_3 : (\mathbb{G}_T \times \{0, 1\}^*)^3 \rightarrow \mathbb{Z}_n^*.$$

Uma chave mestra  $s \xleftarrow{R} \mathbb{Z}_n^*$  também é escolhida, à qual a chave pública  $P_{pub} = sP \in \mathbb{G}_1$  é associada.

O gerador  $g = e(P, Q) \in \mathbb{G}_T$  também é incluso entre os parâmetros públicos do sistema,  $\text{params} = (k, n, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, P, Q, g, P_{pub}, h_0, h_1, h_2, h_3)$ .

- **Set-Secret-Value:** dados  $\text{params}$ , o algoritmo toma  $x_A \xleftarrow{R} \mathbb{Z}_n^*$  como o valor secreto da identidade  $A$ . O usuário  $A$  pode definir  $x_A$ , sua chave parcial privada, independente do algoritmo e, neste caso, será usada como uma senha comum.
- **Set-Public-Value:** dado o valor secreto  $x_A \in \mathbb{Z}_n^*$  da identidade  $A$ , computa  $y_A \leftarrow g^{x_A} \in \mathbb{G}_T$  como o valor público de  $A$ .
- **Private-Key-Extract:** Obtém  $\text{ID}_A \in \{0, 1\}^*$ , o identificador de  $A$  e o valor público  $y_A \in \mathbb{G}_T$ , e calcula a chave privada baseada em identidade de  $A$ ,  $Q_A \leftarrow (h_1(y_A, \text{ID}_A) + s)^{-1}Q \in \mathbb{G}_2$ . A entidade  $A$  consegue verificar a consistência desta chave checando se  $e(h_1(y_A, \text{ID}_A)P + P_{pub}, Q_A) = g$ . Esta configuração é denominada estilo de chave Sakai-Kasahara (SAKAI; KASAHARA, 2003).
- **Set-Private-Key:** dada a chave privada parcial da entidade  $A$ ,  $Q_A \in \mathbb{G}_2$  e o valor secreto  $x_A \in \mathbb{Z}_n^*$ , este algoritmo estabelece o par  $(x_A, Q_A) \in \mathbb{Z}_n^* \times \mathbb{G}_2$

como o par completo da chave privada de  $A$ .

- **Set-Public-Key:** dada a chave privada parcial de  $A$ ,  $Q_A \in \mathbb{G}_2$ , o valor secreto  $x_A \in \mathbb{Z}_n^*$ , e o correspondente valor público  $y_A \in \mathbb{G}_T$ , o assinante toma  $u_A \xleftarrow{R} \mathbb{Z}_n^*$  e calcula

1.  $r_A \leftarrow g^{u_A}$
2.  $h_A \leftarrow h_0(r_A, y_A, \text{ID}_A)$
3.  $T_A \leftarrow (u_A - x_A h_A) Q_A$

A chave pública completa da entidade  $A$  é a tripla  $(y_A, h_A, T_A) \in \mathbb{G}_T \times \mathbb{Z}_n^* \times \mathbb{G}_2$ . Esta configuração é uma combinação da assinatura de Schnorr (sob a chave  $x_A$ ) com a assinatura BLMQ (sob a chave  $Q_A$ ) no valor público  $y_A$  e a identidade  $\text{ID}_A$ .

- **Public-Key-Validate:** dada a chave pública completa da entidade  $A$ ,  $(y_A, h_A, T_A)$ , este algoritmo verifica que  $y_A$  tem ordem  $n$  (i.e. que  $y_A \neq 1$  mas  $y_A^n = 1$ ) e calcula

1.  $r_A \leftarrow e(h_1(y_A, \text{ID}_A)P + P_{\text{pub}}, T_A)y_A^{h_A}$
2.  $v_A \leftarrow h_0(r_A, y_A, \text{ID}_A)$

O verificador aceita a mensagem se, e somente se  $v_A = h_A$ . O processo de validação combina a verificação da assinatura Schnorr com a verificação da assinatura BLMQ.

- **Signcrypt:** Para encriptar  $m \in \{0, 1\}^*$  sob a chave pública do receptor  $y_B \in \mathbb{G}_T$  previamente válida para a identidade  $\text{ID}_B$  e  $P_{\text{pub}}$ , e a chave privada do emissor  $x_A \in \mathbb{Z}_n^*$ , chave pública  $y_A \in \mathbb{G}_T$  e a identidade  $\text{ID}_A$ , o emissor toma  $u \xleftarrow{R} \mathbb{Z}_n^*$  e calcula

1.  $r \leftarrow y_B^u$

2.  $c \leftarrow h_2(r) \oplus m$
3.  $h \leftarrow h_3(r, m, y_A, \text{ID}_A, y_B, \text{ID}_B)$
4.  $z \leftarrow u - x_A h$

O criptograma de assinatura é a tripla  $(c, h, z) \in \{0, 1\}^* \times \mathbb{Z}_n^2$ . Comparado ao método de cifrassinatura de Zheng, as identidades de ambos o emissor e o destinatário são inclusas na equação de autenticação 3, e a equação de assinatura 4 segue o estilo Schnorr em vez do dedicado, porém levemente mais complicado (devido à presença da inversão de campos), estilo Zheng, similar ao DSA (NIST, 2000).

- **Unsigncrypt:** dada a chave pública do emissor  $y_A \in \mathbb{G}_T$  previamente validade para a identidade  $\text{ID}_A$  e  $P_{pub}$ , e a chave privada do receptor  $x_B \in \mathbb{Z}_n^*$ , a chave pública  $y_B \in \mathbb{G}_T$  e a identidade  $\text{ID}_B$ , sob a recepção da tripla  $(c, h, z)$  o receptor verifica se  $h, z \in \mathbb{Z}_n^*$  e calcula

1.  $r \leftarrow y_A^{hx_B} y_B^z$
2.  $m \leftarrow h_2(r) \oplus c$
3.  $v \leftarrow h_3(r, m, y_A, \text{ID}_A, y_B, \text{ID}_B)$

O receptor aceita a mensagem se, e somente se,  $v = h$ . A equação 1 é levemente mais simples que seu correlato em Zheng devido ao estilo Schnorr adotado para a cifrassinatura.

### 5.2.2 Testes de viabilidade

O novo esquema também foi implementado na plataforma JME (*Java Platform Micro Edition*), e testes para validar a viabilidade foram feitos em diversos modelos de aparelhos celulares, além dos emuladores dos ambientes de desenvolvimento *Eclipse* e *NetBeans*.

Os resultados foram satisfatórios, já que os tempos de cifrassinatura e verificação estão de acordo com as métricas estabelecidas e bem mais eficientes em relação ao esquema inicialmente adotado.

O tempo necessário para validar uma chave pública é um pouco maior do que para as demais operações. Porém, conforme observado anteriormente, esta é uma operação que será executada apenas uma vez para cada nova identidade que se deseje validar. A chave validada fica armazenada na memória do aplicativo, não sendo necessário validá-la novamente em uma comunicação futura com o mesmo par.

Os resultados dos testes são apresentados nas tabelas 2 e 3. Foram feitos testes com chaves de 127 e 160 bits, para dois modelos distintos de celulares, Nokia 6275 e Sony Ericsson W200i.

Tabela 2: Testes com o novo esquema (chaves de 127 bits) e comparação com o RSA

Operação	Tempo Nokia 6275 (s)	Tempo Sony Ericsson W200i (s)
Emparelhamento Eta	7,30	2,37
Emparelhamento Ate	7,43	2,38
Private-Key-Extract	2,63	0,93
Check-Private-Key	9,31	2,92
Set-Public-Value	0,66	0,22
Set-Public-Key	3,40	1,15
Public-Key-Validate	10,50	3,35
Signcrypt	0,57	0,21
Unsigncrypt	0,80	0,29
Private RSA-508	1,05	0,39
Public RSA-508	0,03	0,02

### 5.3 Conclusão

Pode-se verificar a partir das tabelas 2 e 3 que os tempos de assinatura e verificação no algoritmo proposto são menores do que os tempos do RSA, para o mesmo nível de segurança.

Tabela 3: Testes com o novo esquema (chaves de 160 bits) e comparação com o RSA

Operação	Tempo Nokia 6275 (s)	Tempo Sony Ericsson W200i (s)
Emparelhamento Eta	10,53	3,59
Emparelhamento Ate	10,54	3,64
Private-Key-Extract	3,72	1,32
Check-Private-Key	12,70	4,46
Set-Public-Value	0,96	0,33
Set-Public-Key	4,96	1,63
Public-Key-Validate	14,94	5,12
Signcrypt	0,77	0,31
Unsigncrypt	1,22	0,45
Private RSA-640	1,85	0,74
Public RSA-640	0,16	0,03

Dado que o tempo de uso do RSA de 1024 bits está no fim, uma nova versão será necessária (KALISKI, 2003). Contudo, para um aumento no nível de segurança do RSA, é preciso aumentar o tamanho das chaves, o que será um impacto razoável nos tempos de assinatura e verificação, uma vez que o aumento é relativamente grande. Em paralelo, um aumento equivalente no nível de segurança do BDCPS, acarreta menor aumento no tamanho das chaves e os impactos nos tempos das operações são menores. Este fato ocorre devido à relação entre o nível de segurança do BDCPS e o tamanho das chaves, que é uma relação diretamente proporcional.

## **6 ESPECIFICAÇÃO DO SISTEMA**

### **6.1 Requisitos funcionais**

### **6.2 Requisitos não-funcionais**

### **6.3 Arquitetura**

### **6.4 Classes**

### **6.5 Casos de uso**

### **6.6 Especificação do protocolo**

O intercâmbio de mensagens entre clientes, ou entre um cliente e a KGB, se dá através do envio de mensagens binárias de SMS. Em nosso protocolo existem 4 tipos de mensagens (4 primitivas). Nesta seção apresentamos como é feita a divisão de bytes em cada tipo de mensagem.<sup>1</sup>

#### **6.6.1 RequestMyQAMessage**

Representa a mensagem que um cliente A envia para KGB contendo sua chave pública  $y_A$ .

---

<sup>1</sup>Os bytes de uma mensagem serão numerados iniciando de 1.

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Param. 1
0x50	0x00	0x00	0x00	Tam. $y_A$	... $y_A$ ...

Figura 1: RequestMyQAMessage

- Byte 1: Byte fixo que identifica uma mensagem do nosso protocolo. Valor 0x50.
- Byte 2: Byte fixo identifica a primitiva RequestMyQAMessage. Valor 0x00.
- Byte 3: Leva o valor do número de bits  $K$  usado na operação, como um inteiro sem sinal.
- Byte 4: Byte reservado para algum possível uso em uma versão futura. Nesta versão tem valor 0x00.
- Byte 5: Armazena um número que informa o comprimento em bytes do parâmetro  $y_A$ .

A partir do byte 6, ocorre o armazenamento dinâmico dos parâmetros. É reservado para cada parâmetro o espaço especificado nos bytes anteriores.

- Parâmetro 1: O  $y_A$ .

### 6.6.2 HerelsYourQAMessage

Representa a mensagem que a KGB envia a um usuário  $A$  sua chave privada parcial  $Q_A$  gerada a partir de  $e$  e encriptada usando a chave pública  $y_A$

do usuário. Somente um usuário em posse do  $x_A$  associado ao  $y_A$  poderá abrir o  $Q_A$  contido nesta mensagem.

- Byte 1: Byte fixo que identifica uma mensagem do nosso protocolo. Valor 0x50.
- Byte 2: Byte fixo identifica a primitiva `HerelsYourQAMessage`. Valor 0x01.
- Byte 3: Leva o valor do número de bits  $K$  usado na operação, como um inteiro sem sinal.
- Byte 4: Byte reservado para algum possível uso em uma versão futura. Nesta versão tem valor 0x00.
- Byte 5: Armazena um número que informa o comprimento em bytes do parâmetro  $c$ .
- Byte 6: Armazena um número que informa o comprimento em bytes do parâmetro  $h$ .
- Byte 7: Armazena um número que informa o comprimento em bytes do parâmetro  $z$ .

A partir do byte 8, ocorre o armazenamento dinâmico dos parâmetros. É reservado para cada parâmetro o espaço especificado nos bytes anteriores.

- Parâmetro 1: Parâmetro  $c$ , parte do criptograma.
- Parâmetro 2: Parâmetro  $h$ , parte do criptograma.
- Parâmetro 3: Parâmetro  $z$ , parte do criptograma.



Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Pa
0x50	0x01	0x00	0x00	Tam. c	Tam. h	Tam. z	...

Figura 2: HerelsYourQAMessage

### 6.6.3 AuthenticationMessage

Representa a mensagem por onde um usuário envia sua chave pública  $y_A$  para um outro usuário. Os parâmetros  $h_A$  e  $t_A$  também são enviados, pois serão usados pelo outro usuário para validar a chave pública  $y_A$  (funcionam quase como um certificado para o  $y_A$ ).

- Byte 1: Byte fixo que identifica uma mensagem do nosso protocolo. Valor 0x50.
- Byte 2: Byte fixo identifica a primitiva AuthenticationMessage. Valor 0x02.
- Byte 3: Leva o valor do número de bits  $K$  usado na operação, como um inteiro sem sinal.
- Byte 4: Byte reservado para algum possível uso em uma versão futura. Nesta versão tem valor 0x00.
- Byte 5: Armazena um número que informa o comprimento em bytes do parâmetro  $y_A$ .
- Byte 6: Armazena um número que informa o comprimento em bytes do parâmetro  $h_A$ .

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Pa
0x50	0x02	0x00	0x00	Tam. $y_A$	Tam. $h_A$	Tam. $T_A$	...

Figura 3: AuthenticationMessage

- Byte 7: Armazena um número que informa o comprimento em bytes do parâmetro  $T_A$ .

A partir do byte 8, ocorre o armazenamento dinâmico dos parâmetros. É reservado para cada parâmetro o espaço especificado nos bytes anteriores.

- Parâmetro 1: Parâmetro  $y_A$ , a chave pública.
- Parâmetro 2: Parâmetro  $h_A$ .
- Parâmetro 3: Parâmetro  $T_A$ .

#### 6.6.4 SigncryptMessage

Representa uma mensagem cifrassinada a ser trocada entre usuários.

- Byte 1: Byte fixo que identifica uma mensagem do nosso protocolo. Valor 0x50.
- Byte 2: Byte fixo identifica a primitiva SigncryptMessage. Valor 0x03.
- Byte 3: Leva o valor do número de bits  $K$  usado na operação, como um inteiro sem sinal.

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Pa
0x50	0x03	0x00	0x00	Tam. c	Tam. h	Tam. z	...

Figura 4: SignCryptedMessage

- Byte 4: Byte reservado para algum possível uso em uma versão futura. Nesta versão tem valor 0x00.
- Byte 5: Armazena um número que informa o comprimento em bytes do parâmetro  $c$ .
- Byte 6: Armazena um número que informa o comprimento em bytes do parâmetro  $h$ .
- Byte 7: Armazena um número que informa o comprimento em bytes do parâmetro  $z$ .

A partir do byte 8, ocorre o armazenamento dinâmico dos parâmetros. É reservado para cada parâmetro o espaço especificado nos bytes anteriores.

- Parâmetro 1: Parâmetro  $c$ , parte do criptograma.
- Parâmetro 2: Parâmetro  $h$ , parte do criptograma.
- Parâmetro 3: Parâmetro  $z$ , parte do criptograma.

## 6.7 Escolha de parâmetros

## **7 IMPLEMENTAÇÃO**

### **7.1 Metodologia**

### **7.2 Ambiente de desenvolvimento**

### **7.3 Bibliotecas usadas**

### **7.4 Problemas encontrados**

## **8 RESULTADOS**

### **8.1 Desempenho**

### **8.2 Comparação com outras soluções**

## **9 CONCLUSÃO**

### **9.1 Análise dos resultados**

### **9.2 Possíveis desenvolvimentos futuros**

## REFERÊNCIAS

BARRETO, P. S. L. M. et al. *Toward Efficient Certificateless Signcryption from (and without) Bilinear Pairings*. Junho 2008. Preprint.

\_\_\_\_\_. Efficient and provably-secure identity-based signatures and signcryption from bilinear maps. In: *Advanced in Cryptology – Asiacrypt'2005*. [S.l.]: Springer, 2005. (Lecture Notes in Computer Science, v. 3788), p. 515–532.

BONEH, D.; FRANKLIN, M. Identity-based encryption from the Weil pairing. In: *Advanced in Cryptology – Crypto'2001*. [S.l.]: Springer, 2001. (Lecture Notes in Computer Science, v. 2139), p. 213–229.

ENCK, W. et al. Exploiting open functionality in sms-capable cellular networks. In: *Proceedings of the 12th ACM conference on Computer and communications security*. New York, NY, USA: ACM Press, 2005. p. 393–404.

FREEMAN, D.; SCOTT, M.; TESKE, E. *A Taxonomy of Pairing-Friendly Elliptic Curves*. 2006. IACR ePrint Archive, report 2006/372. <http://eprint.iacr.org/2006/372>.

KALISKI, B. *TWIRL and RSA Key Size*. Maio 2003. <http://www.rsa.com/rsalabs/node.asp?id=2004>.

NG, Y. L. *Short Message Service (SMS) Security Solution for Mobile Devices*. Monterey, California, USA: [s.n.], 2006. 17–19 p.

NIST. *Federal Information Processing Standard (FIPS 186-2) – Digital Signature Standard (DSS)*. [S.l.], January 2000.

ORTIZ, C. *The Wireless Messaging API*. December 2002. Sun Developer Network (SDN) article. <http://developers.sun.com/mobility/midp/articles/wma/index.html>.

SAKAI, R.; KASAHARA, M. ID based cryptosystems with pairing on elliptic curve. In: *SCIS'2003*. Hamamatsu, Japan: [s.n.], 2003.

SCHNORR, C. P. Efficient signature generation by smart cards. *Journal of Cryptology*, v. 4, n. 3, p. 161–174, 1991.

SHAMIR, A. Identity based cryptosystems and signature schemes. In: *Advances in Cryptology – Crypto'84*. [S.l.]: Springer, 1984. (Lecture Notes in Computer Science, v. 0196), p. 47–53.

ZHENG, Y. Digital signcryption or how to achieve  $\text{cost}(\text{signature} \ \& \ \text{encryption})$   
«  $\text{cost}(\text{signature}) + \text{cost}(\text{encryption})$ . In: *Advanced in Cryptology – Crypto'97*.  
[S.l.]: Springer, 1997. (Lecture Notes in Computer Science, v. 1294), p.  
165–179.



## **APÊNDICE A - APÊNDICE**