

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

SISTEMA DE SMS SEGURO

São Paulo
2008

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

SISTEMA DE SMS SEGURO

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**

SISTEMA DE SMS SEGURO

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

São Paulo
2008

AGRADECIMENTOS

Ao nosso orientador Prof. Dr. Paulo Sérgio Licciardi Messeder Barreto, pelo envolvimento e constante incentivo ao projeto.

Aos nossos familiares, por fornecer o suporte necessário para alcançarmos nossos objetivos e por compreenderem nossa constante ausência nos últimos meses.

Aos nossos professores, por guiar-nos na busca pelo conhecimento necessário à realização do trabalho.

Aos nossos colegas, que sempre nos apoiaram e ajudaram a superar as dificuldades enfrentadas durante o curso.

Ao André Felipe Santos, por auxiliar-nos na realização dos testes finais.

Ao Felipe Sanches, por ter criado o ícone do aplicativo.

RESUMO

Este trabalho consistiu na especificação, projeto e implementação de um sistema que garante serviços de segurança na troca de mensagens *SMS* entre aparelhos de telefonia celular.

A necessidade da implementação de mais uma camada de segurança sobre a rede de telefonia celular *GSM* justifica-se pelo fato de as mensagens trafegarem pela rede interna da operadora sem qualquer mecanismo de segurança, deixando-a vulnerável a quaisquer atacantes que possam comprometer o sistema e obter acesso à sua base de dados *SMS*, incluindo funcionários mal intencionados. Desse modo, aplicações que requeiram segurança ficam impossibilitadas de se aproveitar das qualidades deste meio: mobilidade, leveza e baixo custo.

Os paradigmas de segurança atualmente empregados na *Internet* não são adequados a esse cenário, uma vez que há grandes limitações de banda e poder de processamento. Analisamos conceitos relativamente recentes de Criptografia de Curvas Elípticas e Criptografia Baseada em Identidades, chegando a um esquema híbrido entre essas técnicas e a criptografia de chave pública convencional que atende aos requisitos estabelecidos.

ABSTRACT

This work consisted of the specification, design and implementation of a system that guarantees security services to the mobile Short Message Service (SMS).

The need of an extra security layer onto the *GSM* network resides on the fact that short messages are exchanged through the carrier's internal network as plain text, vulnerable to any attacker that manages to compromise the system and gain access to the SMS database, including malicious employees. Therefore, security-sensitive applications remain unable to take advantage from this media's best features: mobility, lightness and low cost.

The security paradigms currently adopted on the Internet don't fulfill this background's requirements, since there are strong bandwidth and processing restrictions. The analysis of the relatively recent concepts of Elliptic Curve Cryptography and Identity-Based Cryptography lead us to a hybrid scheme that assembles the best qualities of these techniques and Public Key Cryptography and fulfills the established requirements.

SUMÁRIO

Lista de Figuras

Lista de Tabelas

Lista de abreviaturas e siglas

1	Introdução	15
1.1	Cenário	16
1.1.1	Alternativas existentes	16
1.1.2	Ambiente	17
1.2	Objetivos	18
1.3	Metodologia	19
1.4	Organização	19
2	Preliminares teóricas	21
2.1	Serviços de Segurança	21
2.2	Criptografia de Chave Simétrica	22
2.2.1	Introdução	22
2.2.2	Administração e distribuição de chaves	22
2.3	Criptografia de chave pública	24
2.3.1	Introdução	24

2.3.2	Confidencialidade	24
2.3.3	Irretratabilidade	25
2.4	Logaritmos Discretos	26
2.4.1	Problemas Baseados em Logaritmos Discretos	26
2.4.2	Geração de chaves com Logaritmos Discretos	27
2.5	Curvas Elípticas	27
2.5.1	Histórico	27
2.5.2	Grupos	28
2.5.3	Grupos em Curvas Elípticas	30
2.5.4	Geração de chaves em curva elíptica	31
2.6	Criptografia baseada em identidades e sistemas isentos de certificados	32
2.7	Emparelhamentos	32
2.8	BLMQ	33
2.9	BDCPS	34
3	Discussão	38
3.1	Escopo	38
3.2	Métricas e Restrições	39
4	Análise de requisitos do sistema	42
4.1	Requisitos funcionais	42
4.1.1	Cifrassinatura de mensagem	42

4.1.2	Verificação de mensagem	42
4.1.3	Envio de mensagem	43
4.1.4	Geração de chave privada	43
4.2	Requisitos não-funcionais	43
4.2.1	Usabilidade	43
4.2.2	Desempenho	44
4.2.3	Confiabilidade	44
4.2.4	Disponibilidade	44
4.2.5	Compatibilidade	44
4.2.6	Portabilidade	44
4.2.7	Segurança	45
4.3	Casos de uso	45
4.3.1	Cadastrar-se no sistema	45
4.3.2	Autenticar novo contato	47
4.3.3	Enviar Mensagem	48
4.3.4	Recepção de Mensagem	49
4.3.5	Encriptar/Assinar Mensagem	50
4.3.6	Decriptar/Verificar Mensagem	51
5	Escolha do esquema criptográfico	52
5.1	Por que criptografia em curvas elípticas?	52
5.2	BLMQ	53

5.2.1	Testes de viabilidade	54
5.3	BDCPS	55
5.3.1	Vantagens do esquema	55
5.3.2	Testes de viabilidade	56
5.4	Análise dos resultados preliminares	57
6	Especificação e projeto	58
6.1	Arquitetura	58
6.2	Classes	59
6.2.1	Descrição	59
6.3	Especificação do protocolo de troca de mensagens	59
6.3.1	SignupMessage	60
6.3.2	SignupResponse	61
6.3.3	ValidationMessage	63
6.3.4	SigncryptedException	65
7	Implementação	67
7.1	Ambiente de desenvolvimento	67
7.2	Bibliotecas utilizadas	67
7.2.1	SMSPairings	67
7.2.2	BouncyCastle	68
7.2.3	Floggy	68
7.3	Escolha de parâmetros	68

7.3.1	Escolha do tamanho de chave	68
7.3.2	Escolha da porta SMS	69
7.4	Telas do sistema	70
8	Resultados	71
8.1	Desempenho	71
8.2	Testes entre operadoras	72
9	Conclusão	73
9.1	Análise dos resultados	73
9.2	Perspectivas futuras	74
9.3	Considerações finais	75
	Referências	76
	Apêndice A – Glossário	78
	Apêndice B – Diagramas de classe detalhados	79
	Apêndice C – Telas do sistema	84
	Apêndice D – Desempenho da implementação final	91

LISTA DE FIGURAS

1	Canal inseguro	17
2	Soma em Curva Elíptica	31
3	Diagrama de casos de uso do sistema	46
4	Diagrama de implantação do sistema	58
5	Diagrama de classes do sistema	60
6	Estabelecimento da chave privada com a autoridade certificadora	61
7	Fluxo de comunicação entre dois usuários	62
8	SignupMessage	62
9	SignupResponse	63
10	ValidationMessage	64
11	SigncryptedException	66
12	Diagrama de classes do pacote Application	80
13	Diagrama de classes do pacote Data	81
14	Diagrama de classes do pacote Messaging	82
15	Diagrama de classes do pacote Protocol	83
16	Tela inicial do sistema	84
17	Tela de primeiro uso do sistema	85
18	Tela de validação de contatos	86
19	Tela de lista de contatos	87

20	Tela de envio de mensagem	88
21	Tela de lista de mensagens	89
22	Tela da caixa de entrada	90
23	Gráfico: tempo de Signcrypt x tamanho da mensagem	92
24	Gráfico: tempo das operações x tamanho da chave (Nokia E51)	93
25	Gráfico: tempo das operações x tamanho da chave (Nokia 6275)	94
26	Gráfico: tempo das operações x tamanho da chave (Emulador WTK2.5.2)	95

LISTA DE TABELAS

1	Testes com BLMQ	54
2	Testes com o novo esquema (chaves de 127 bits) e comparação com o RSA	57
3	Testes com o novo esquema (chaves de 160 bits) e comparação com o RSA	57
4	Testes com a implementação final (chaves de 176 bits)	71
5	Compatibilidade entre operadoras	72
6	Medida de tempo de Signcrypt x tamanho da mensagem	91

LISTA DE ABREVIATURAS E SIGLAS

AES - *Advanced Encryption Standard*

API - *Applicaition Programming Interface*

CLDC - *Connected Limited Device Configuration*

CMA - *Chosen-message attack*

DES - *Data Encryption Standard*

DSA - *Digital Signature Algorithm*

DSS - *Digital Signature System*

DL - *Discrete Logarithm*

ECC - *Elliptic Curve Cryptography*

ECDLP - *Elliptic Curve Discrete Logarithm Problem*

EU - *Existential unforgeability*

GSM - *Global System for Mobile Comunnication*

IBS - *ID-based signature*

IDE - *Integrated Development Environment*

J2ME - *Java 2, Micro Edition*

KGB - *Key Generation Bureau*

MAC - *Message Authentication Code*

MTTF - *Mean Time To Fail*

MIDP - *Mobile Information Device Profile*

PKI - *Private Key Infrastructure*

RC4 - *Rivest Cipher 4*

RSA - *Rivest, Shamir, Adleman*

SMS - *Short Message Service*

SMSC - *Short Message Service Center*

SMTP - *Simple Mail Transfer Protocol*

VPN - *Virtual Private Network*

WMA - *Wireless Messaging API*

WTK - *Wireless Toolkit*

1 INTRODUÇÃO

O Serviço de Mensagens Curtas (*Short Message Service*, ou SMS) é um serviço oferecido por operadoras de telefonia celular para que seus usuários troquem mensagens curtas de texto com outros usuários da rede ou com serviços da Internet. Atualmente, cerca de 2.4 bilhões de pessoas utilizam o SMS no mundo.

Comercialmente, as mensagens SMS moveram uma massiva indústria em 2006, com cerca de 81 bilhões de dólares no mundo. Em 2010 cerca de 2.3 trilhões de mensagens de texto serão enviadas no mundo, gerando 72.5 bilhões de dólares para as operadoras de celular, de acordo com previsões realizadas no ano de 2006 pela Gartner Dataquest. A maioria deles transformam-se em lucro, porque a margem de lucro em mensagens de texto flutua em cerca de 90%, mais que o dobro do que as operadoras obtêm com serviços de voz. (SYLVERS, 2007)

A rede GSM (Global System for Mobile Communication), sobre a qual as mensagens SMS trafegam, usa o mecanismo *store-and-forward*, que é similar ao serviço SMTP de correio eletrônico. Em vez de servidores de *e-mail*, são usados centros de SMS (SMSC) que armazenam as mensagens SMS antes de serem enviadas para o fornecedor de serviços (operadora) ou para outro SMSC.

Embora as conexões entre um SMSC e os nós da rede GSM sejam pro-

tegidas por túneis VPN, as mensagens SMS ficam armazenadas em texto claro no SMSC. Isto significa que os operadores ou um atacante que invada o sistema podem visualizar e alterar o conteúdo de todas as mensagens SMS que estão armazenadas no SMSC, além de enviar mensagens em nome de outrem (NG, 2006).

Além de comprometer a privacidade dos usuários, essa vulnerabilidade da rede GSM limita usos da tecnologia SMS além da comunicação interpessoal, como a realização de transações bancárias, sistemas de comunicação que requeiram confidencialidade e integridade (órgãos militares e governamentais, comunicação corporativa) ou ainda serviços de monitoração remota de dados sensíveis (ENCK et al., 2005).

Desse modo, nos motivamos a projetar e implementar um sistema que oferecesse serviços de segurança à plataforma SMS de maneira transparente, isto é, sem que fossem necessárias mudanças na rede atual e, por outro lado, fosse viável em dispositivos móveis, dada suas limitações de banda, processamento e energia.

1.1 Cenário

1.1.1 Alternativas existentes

Atualmente não existem soluções universalmente adotadas para garantir segurança em mensagens SMS. Na maioria das transações 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.

No início de nossa pesquisa, as alternativas de sistemas de segurança pra SMS disponíveis eram escassas. No decorrer do ano, diversas novas

soluções foram surgindo.

1.1.2 Ambiente

Na figura 1, as entidades A (Alice) e B (Bob) estão se comunicando sobre um canal inseguro. Assumimos que todas as comunicações têm a presença de um agressor E (Eve) cujo objetivo é explorar falhas nos serviços de segurança sendo fornecidos por A e B .

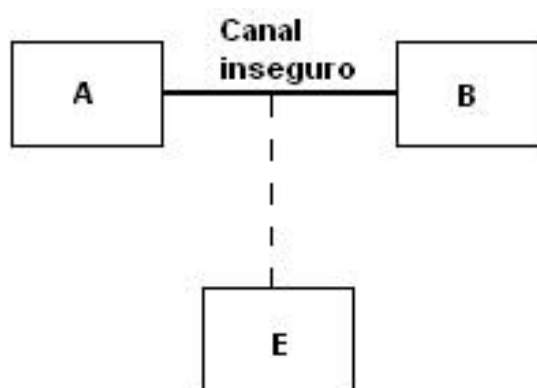


Figura 1: Canal inseguro

Por exemplo, A e B poderiam ser 2 pessoas comunicando-se sobre uma rede de telefonia celular, e E está tentando interceptar a comunicação.

Ou, \tilde{A} poderia ser um *web browser* de um indivíduo A que está em processo de compra de um produto de uma loja *on-line* \tilde{B} representada por seu *site* B . Neste cenário, o canal de comunicações é a *Internet*. Um agressor E poderia tentar ler o tráfego entre A e B , portanto saber a informação sobre o cartão de crédito de A , ou poderia tentar personificar A ou B na transação. Como um terceiro exemplo, considere a situação onde A está enviando uma mensagem via *e-mail* para B sobre a Internet. Um agressor E poderia tentar ler a mensagem, modificar pedaços selecionados, ou personificar A enviando mensagens dela mesma para B . Finalmente, considere o cenário onde A é um smart card que está em processo de autenticar seu possuidor \tilde{A} em um

computador *mainframe* *B* em uma sala protegida do banco. Aqui, *E* poderia tentar monitorar as comunicações para obter informações sobre a conta de *Ã*, ou poderia tentar personificar *Ã* para sacar fundos da conta de *Ã*. Deveria ser evidente destes exemplos que uma entidade se comunicando não é necessariamente um humano, mas pode ser um computador, *smart card*, ou um módulo de software agindo no lugar de um indivíduo ou uma organização tal como uma loja ou um banco.(MENEZES; OORSCHOT; VANSTONE, 1996)

1.2 Objetivos

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 ¹ (ORTIZ, 2002). Além disto, existem limitações de processamento no dispositivo celular, que podem comprometer a usabilidade de um esquema criptográfico tradicional (NG, 2006).

Devido a estas dificuldades tecnológicas, o cenário atual não apresenta uma grande variedade de soluções abrangendo os problemas de segurança supracitados. Desse modo, nos propomos a projetar, e implantar um sistema capaz de prover confidencialidade, integridade e autenticidade a mensagens SMS (*Short Message Service*) sem extrapolar as limitações de recursos computacionais e de ocupação de banda típicas desse ambiente.

¹Bytes de 8 bits. Essa diferenciação é feita pois o protocolo SMS original trabalha com bytes de 7 bits.

1.3 Metodologia

Nossa metodologia de pesquisa se dividiu basicamente em três vertentes. A primeira consistiu no estudo do cenário, o detalhamento do problema e o levantamento das necessidades, além da especificação de uma solução que as endereçasse.

Posteriormente, realizamos o estudo de esquemas de criptografia de modo a buscar o que mais se adequasse aos requisitos e limitações do meio. Esse estudo incluiu a realização de testes em microcomputadores convencionais e em dispositivos móveis com pseudo-implementações de diversos esquemas com o objetivo de comparar seu desempenho em termos de necessidades de processamento. Como veremos, nenhum esquema pesquisado atendeu a nossos requisitos, o que motivou o desenvolvimento de um novo protocolo de segurança totalmente voltado a estes.

Por último, realizamos a implementação do sistema conforme sua especificação e realizamos testes de modo a confrontar os dados obtidos em um ambiente real com as previsões feitas na fase anterior.

1.4 Organização

O restante desta monografia organiza-se como se segue. No capítulo 2 apresentamos algumas preliminares teóricas necessárias ao desenvolvimento de nossa pesquisa e essenciais ao entendimento deste trabalho. O capítulo 3 aprofunda a discussão do tema, delineando o escopo do sistema e definindo métricas de avaliação das soluções estudadas. Os capítulos 4 a 7 apresentam a análise de requisitos e dos casos de uso do sistema e justificam as decisões de projeto e implementação tomadas com base nessa análise. Em seguida,

apresentamos os resultados obtidos e finalmente concluimos o trabalho.

2 PRELIMINARES TEÓRICAS

2.1 Serviços de Segurança

Listamos abaixo alguns dos principais serviços de segurança da informação:

- **Confidencialidade:** manter secretos os dados de todos a não ser àqueles autorizados a acessá-los - mensagens enviadas por A para B não devem ser legíveis por E .
- **Integridade:** assegurar que os dados não sejam alterados por entidades não autorizadas - B deve ser capaz de detectar quando dados enviados por A tenham sido modificados acidentalmente ou deliberadamente por um atacante E .
- **Autenticidade dos dados:** confirmar a fonte dos dados - B deve ser capaz de verificar que dados supostamente enviados por A de fato foram originados por A .
- **Autenticidade da entidade:** confirmar a identidade de determinada entidade: B deve poder convencer-se da veracidade da identidade de A .
- **Irretratabilidade:** prevenir uma entidade de negar compromettimentos ou atos anteriores - quando B recebe uma mensagem supostamente de A , não apenas B está convencido de que a mensagem se originou em A ,

mas B pode convencer uma terceira parte disso; portanto A não pode negar ter enviado a mensagem para B . Algumas aplicações podem ter outros objetivos de segurança tais como anonimato das entidades em comunicação ou controle de acesso (a restrição de acessar recursos).

2.2 Criptografia de Chave Simétrica

2.2.1 Introdução

Os sistemas criptográficos podem ser amplamente divididos em dois tipos. Em esquemas de chave simétrica, as entidades em comunicação compartilham uma informação, usada como chave, que é ao mesmo tempo secreta e autêntica. Consequentemente, eles podem usar um esquema de encriptação simétrica tal como o *Data Encryption Standard (DES)*, *RC4*, ou o *Advanced Encryption Standard (AES)* para prover o serviço de confidencialidade.

Eles também podem usar um algoritmo de código de autenticação de mensagens, tal como o *HMAC*, para reunir os serviços de integridade e autenticação da origem dos dados.

Por exemplo, se confidencialidade fosse desejada e a chave secreta compartilhada entre A e B fosse k , então A encriptaria uma mensagem m , em texto claro, usando uma função de encriptação ENC e a chave k e transmitiria a cifra resultante $c = ENC_k(m)$ para B . Ao receber c , B usaria a função de deciptação DEC e a mesma chave k para recuperar $m = DEC_k(c)$.

2.2.2 Administração e distribuição de chaves

A principal vantagem da criptografia de chave simétrica é a alta eficiência. Contudo, há significantes desvantagens nestes sistemas. Uma delas é o

conhecido problema da distribuição de chaves - a necessidade de um canal que seja secreto e autenticado para a distribuição das chaves. Em algumas aplicações, esta distribuição pode ser convenientemente feita por usar um canal fisicamente seguro, tal como um emissário de confiança. Outra maneira é usar os serviços de uma terceira parte confiável *on-line* que inicialmente estabelece chaves secretas com todas as entidades na rede. Essa entidade usa essas chaves para distribuir as informações de chaves para as entidades em comunicação quando requerido¹. Soluções como esta podem ser bem apropriadas para ambientes onde há uma autoridade central aceitável e confiável, mas é claramente intratável em aplicações tal como *e-mail* na *Internet*.

Uma segunda desvantagem é o problema de administração de chaves. Em uma rede de N entidades, cada entidade pode ter que manter diferentes informações de chaves com cada uma das $N - 1$ entidades. Logo, seriam necessárias $N(N - 1)/2$ chaves privadas em toda a rede, o que inviabiliza a administração ao passo que N se torna grande. Este problema pode ser aliviado usando serviços de uma terceira parte *on-line* que distribui as chaves conforme são requeridas, assim reduzindo a necessidade das entidades de armazenar múltiplas chaves seguramente. Novamente, contudo, tais soluções não são práticas em alguns cenários. Finalmente, uma vez que a informação sobre as chaves é compartilhada entre duas (ou mais) entidades, técnicas de chave simétrica não podem ser usadas para implementar esquemas de assinatura digital (DSS) elegantes que forneçam serviços de irretratabilidade. Isto porque é impossível distinguir entre as ações tomadas por diferentes detentores de uma chave secreta².(MENEZES; OORSCHOT; VANSTONE, 1996)

¹Este modo de usar uma terceira parte centralizada para distribuir chaves para algoritmos de chave simétrica às partes conforme elas necessitarem é usado pelo protocolo de autenticação da rede *Kerberos* para aplicações cliente/servidor.

²Esquemas de assinaturas digitais podem ser implementados usando técnicas de chave simétrica; contudo, estes esquemas geralmente são geralmente impraticáveis quando for requerido o uso de uma terceira parte confiável *on-line* ou de novas chaves para cada assinatura.

2.3 Criptografia de chave pública

2.3.1 Introdução

A noção de criptografia de chave pública foi introduzida por Diffie e Hellman (DIFFIE; HELLMAN, 1976) e Merkle (MERKLE, 1978) para resolver as deficiências da criptografia de chaves simétricas mencionadas anteriormente. Em contraste aos esquemas de chave simétrica, os esquemas de chave pública requerem apenas que as entidades em comunicação troquem informações de chaves que são autênticas (mas não secretas). Cada entidade seleciona um único par (e, d) consistindo de uma chave pública e , e uma chave privada relacionada d que a entidade mantém secreta). As chaves têm a propriedade de que é computacionalmente impraticável determinar a chave privada apenas de conhecimento da chave pública.

2.3.2 Confidencialidade

Se a entidade A deseja enviar uma mensagem confidencial m para uma entidade B , ela obtém uma cópia autêntica da chave pública de B e_B , e usa a função de encriptação ENC de uma esquema de chave pública para computar a cifra $c = ENC_{e_B}(m)$. A então transmite c para B , que usa a função de decriptação DEC e sua chave privada d_B para recuperar a mensagem clara: $m = DEC_{d_B}(c)$. A presunção é que um agressor com posse apenas de e_B (mas não de d_B) não consegue decriptar c . Observe-se que não há nenhuma necessidade de descrição de e_B . É essencial apenas que A obtenha uma cópia autêntica de e_B - por outro lado A encriptaria m usando a chave pública e_E de alguma entidade E tentando personificar B , e m seria recuperável por E .

2.3.3 Irretratabilidade

Esquemas de assinatura digital podem ser desenvolvidos para autenticação da origem e integridade dos dados, e para facilitar o fornecimento de serviços de irretratabilidade. Uma entidade A usaria o algoritmo de geração de assinatura $SIGN$ de um esquema de assinatura digital e sua chave privada d_A para computar a assinatura da mensagem: $s = SIGN_{d_A}(m)$. Ao receber m e s , uma entidade B que tem uma cópia autêntica da chave pública de A e_A usa um algoritmo de verificação de assinatura para confirmar que s foi de fato gerado a partir de m e d_A . Uma vez que d_A é presumivelmente conhecido por A , B está seguro de que a mensagem foi realmente originada por A . Ademais, uma vez que a verificação requer apenas quantidades não secretas m e e_A , a assinatura s para m pode também ser verificada por uma terceira parte que poderia estabelecer contestações se A negar ter assinado a mensagem m . Diferente das assinaturas escritas à mão, a assinatura s de A depende da mensagem m sendo assinada, prevenindo um forjador de simplesmente acrescentar s a uma mensagem \tilde{m} linha e afirmar que A assinou \tilde{m} . Mesmo embora não haja nenhuma necessidade de segredo com relação à chave pública e_A , é essencial que os verificadores usem uma cópia autêntica de e_A quando verificarem assinaturas geradas por A .

Deste modo, a criptografia de chave pública fornece soluções elegantes para os três problemas com criptografia de chave simétrica: distribuição de chaves, administração de chaves e suporte à irretratabilidade. Deve-se notar que embora necessidade de um canal secreto para distribuição de chaves tenha sido eliminado, implementar uma infra-estrutura de chave pública (*Public Key Infrastructure*, ou PKI) para distribuir e administrar chaves públicas pode ser um desafio formidável na prática. Também, operações em chave pública em geral são significativamente mais lentas do que seus respectivos

na criptografia de chave simétrica. Portanto, sistemas híbridos que se beneficiam da eficiência dos algoritmos de chave simétrica e da funcionalidade dos algoritmos de chave pública são frequentemente usados.

Em um esquema de chave pública, um par de chaves é selecionado para que o problema de calcular a chave privada a partir da chave pública seja equivalente a resolver um problema computacional considerado intratável. Os problemas teóricos numéricos cuja intratabilidade constrói a base para a segurança dos esquemas comumente usados são:

- O problema da **fatoração de inteiros**, cuja dificuldade é essencial para a segurança da encriptação *RSA* e esquemas de assinatura.
- O problema do **logaritmo discreto**, cuja dificuldade é essencial para a segurança da encriptação de chave pública *ElGamal* e esquemas de assinatura e suas variantes tais como o *Digital Signature Algorithm* (DSA).
- O problema do **logaritmo discreto em curvas elípticas**, cuja dificuldade é essencial para a segurança de todos os esquemas baseados em curvas elípticas.(MENEZES; OORSCHOT; VANSTONE, 1996)

2.4 Logaritmos Discretos

2.4.1 Problemas Baseados em Logaritmos Discretos

O primeiro sistema baseado em logaritmo discreto foi o protocolo de troca de chaves proposto por Diffie e Hellman em 1976 (DIFFIE; HELLMAN, 1976). Em 1984, ElGamal descreveu a encriptação de chave pública baseada no problema do logaritmo discreto e esquemas de assinatura (ELGAMAL, 1985). Desde então, muitas variantes destes esquemas foram propostas.

2.4.2 Geração de chaves com Logaritmos Discretos

Em sistemas de logaritmos discretos, um par de chaves está associado com um conjunto de parâmetros públicos do domínio (p, q, g) . Aqui, p é um primo, q é um divisor primo de $p - 1$, e $g \in [1, p - 1]$ tem ordem q (i.e., $t = q$ é o menor inteiro positivo satisfazendo $g^t \equiv 1 \pmod{p}$). Uma chave privada é um inteiro x que é selecionado uniformemente de modo aleatório no intervalo $[1, q - 1]$ (esta operação é denotada $x \in [1, q - 1]$, e a chave pública correspondente é $y = g^x \bmod p$. O problema de determinar x dados os parâmetros do domínio (p, q, g) e y é o problema do logaritmo discreto (DLP). (MENEZES; OORSCHOT; VANSTONE, 1996)

2.5 Curvas Elípticas

2.5.1 Histórico

O estudo das curvas elípticas por matemáticos data da metade do século XIX. Em 1984, Hendrik Lenstra (LENSTRA, 1987) descreve um engenhoso algoritmo para fatorar inteiros que recai nas propriedades das curvas elípticas. Esta descoberta motivou os pesquisadores a investigar novas aplicações em criptografia sobre curvas elípticas e teoria computacional dos números.

A criptografia de chave pública foi concebida em 1976, mas sua primeira construção prática se seguiu em 1977 quando Ron Rivest, Adi Shamir e Len Adleman propuseram o protocolo agora tão conhecido RSA (RIVEST; SHAMIR; ADLEMAN, 1978) cuja segurança é baseada na intratabilidade do problema da fatoração inteira. A criptografia baseada em curvas elípticas (ECC) foi descoberta em 1985 por Neal Koblitz e Victor Miller.

Os protocolos de ECC são baseados em um problemas mais difícil de

resolver que o do RSA, o ECDLP, sendo que os melhores algoritmos para resolvê-lo levam tempo exponencial frente ao tempo sub-exponencial para o RSA. Isso acarreta em um maior nível de segurança para chaves de menor tamanho comparativamente.

No fim dos anos 90, sistemas sobre curvas elípticas começaram a receber aceitação comercial quando organizações de padrões respeitadas especificaram protocolos sobre curvas elípticas, e empresas privadas incluíram estes protocolos nos seus produtos de segurança. Hoje, ECC é considerado o estado-da-arte em criptografia de chave pública.

Os sistemas baseados em logaritmos discretos apresentados anteriormente podem ser descritos na configuração de um grupo cíclico finito. A definição de grupos segue abaixo.

2.5.2 Grupos

Um grupo abeliano $(\mathbb{G}, *)$ consiste de um conjunto \mathbb{G} com uma operação binária $*$: $\mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}$ satisfazendo as seguintes propriedades: (Associatividade) $a * (b * c) = (a * b) * c$ para todos os $a, b, c \in \mathbb{G}$ (Existência de uma identidade) Existe um elemento $e \in \mathbb{G}$ tal que $a * e = e * a = a$ para todo $a \in \mathbb{G}$. (Existência de inversos) Para cada $a \in \mathbb{G}$, existe um elemento $b \in \mathbb{G}$, chamado inverso de a , tal que $a * b = b * a = e$. (Comutatividade) $a * b = b * a$ para todos $a, b \in \mathbb{G}$.

A operação do grupo é geralmente chamada de adição (+) ou multiplicação (*). Em primeira instância, o grupo é chamado de grupo aditivo, o elemento (aditivo) identidade é normalmente denotado por 0, e o inverso (aditivo) d e a é denotado por $-a$. Em uma segunda instância, o grupo é chamado de grupo multiplicativo, o elemento (multiplicativo) identidade é denotado por 1

e o inverso (multiplicativo) de a é denotado por a^{-1} . O grupo é finito se \mathbb{G} é um conjunto finito, no caso em que o número de elementos em \mathbb{G} é chamado a ordem de \mathbb{G} . Por exemplo, seja p um número primo, e $F_p = 0, 1, 2, \dots, p-1$ denota o conjunto dos inteiros módulo p . Então $(F_p, +)$, onde a operação $+$ é definida com a operação de adição de inteiros módulo p , é um grupo finito aditivo de ordem p com elemento identidade (aditivo) 0 . Além disso, (F_p^*, \bullet) , onde F_p^* denota os elementos diferentes de zero em F_p e a operação \bullet é definida como a multiplicação de inteiros módulo p , é um grupo finito multiplicativo de ordem $p-1$ com elemento identidade (multiplicativo) 1 . A tripla $(F_p, +, \bullet)$ é um corpo finito, denotado mais sucintamente por F_p .

Agora, se \mathbb{G} é um grupo finito multiplicativo de ordem n e $g \in \mathbb{G}$, então o menor inteiro positivo t tal que $gt = 1$ é chamado de ordem de g ; esse t sempre existe e é divisor de n . O conjunto $\langle g \rangle = \{g^i : 0 \leq i \leq t-1\}$ de todas as potências de g é ele próprio um grupo sobre a mesma operação como \mathbb{G} , e é chamado um subgrupo cíclico de \mathbb{G} gerado por g . Declarações análogas são verdadeiras se \mathbb{G} é escrito aditivamente. Assim, a ordem de $g \in \mathbb{G}$ é o menor divisor positivo t de n tal que $tg = 0$, e $\langle g \rangle = \{ig : 0 \leq i \leq t-1\}$. Aqui, tg denota o elemento obtido por adicionar t cópias de g . Se G tem um elemento g de ordem n , então G é dito ser um grupo cíclico e g é um gerador de \mathbb{G} . Por exemplo, com os parâmetros do DL (p, q, g) definidos anteriormente, o grupo multiplicativo (F_p^*, \bullet) é um grupo cíclico de ordem $p-1$. Ademais, $\langle g \rangle$ é um subgrupo cíclico de ordem q .

2.5.3 Grupos em Curvas Elípticas

Seja p um número primo, e F_p o corpo dos inteiros módulo p . Uma curva elíptica E sobre F_p é definida por uma equação da forma

$$y^2 = x^3 + ax + b \quad (2.1)$$

onde $a, b \in F_p$ satisfaz $4a^3 + 27b^2 \neq 0 \pmod{p}$. Um par (x, y) , onde $x, y \in F_p$, é um ponto na curva se (x, y) satisfaz a equação 2.1. O ponto no infinito, denotado por ∞ , também é considerado estar contido na curva. O conjunto de todos os pontos sobre E é denotado por $E(F_p)$. Por exemplo, se E é uma curva elíptica sobre F_7 definida pela equação:

$$y^2 = x^3 + 2x + 4 \quad (2.2)$$

Então, os pontos sobre E são:

$$E(F_7) = \infty, (0, 2), (0, 5), (1, 0), (2, 3), (2, 4), (3, 3), (3, 4), (6, 1), (6, 6).$$

Agora, há um método bem conhecido para somar 2 pontos numa curva elíptica $P : (x_1, y_1)$ e $Q : (x_2, y_2)$ para produzir um terceiro ponto na curva $R : (x_3, y_3)$, conforme ilustrado na Figura 2 (MENEZES; OORSCHOT; VANSTONE, 1996).

A regra de adição requer algumas operações aritméticas (adição, subtração, multiplicação e inversão) em F_p com as coordenadas x_1, y_1, x_2, y_2 . Com esta regra de adição, o conjunto dos pontos $E(F_p)$ forma um grupo abeliano (aditivo) com ∞ servindo como elemento neutro. Subgrupos cíclicos destes grupos sobre curvas elípticas podem ser agora usados para implementar sistemas de logaritmos discretos.

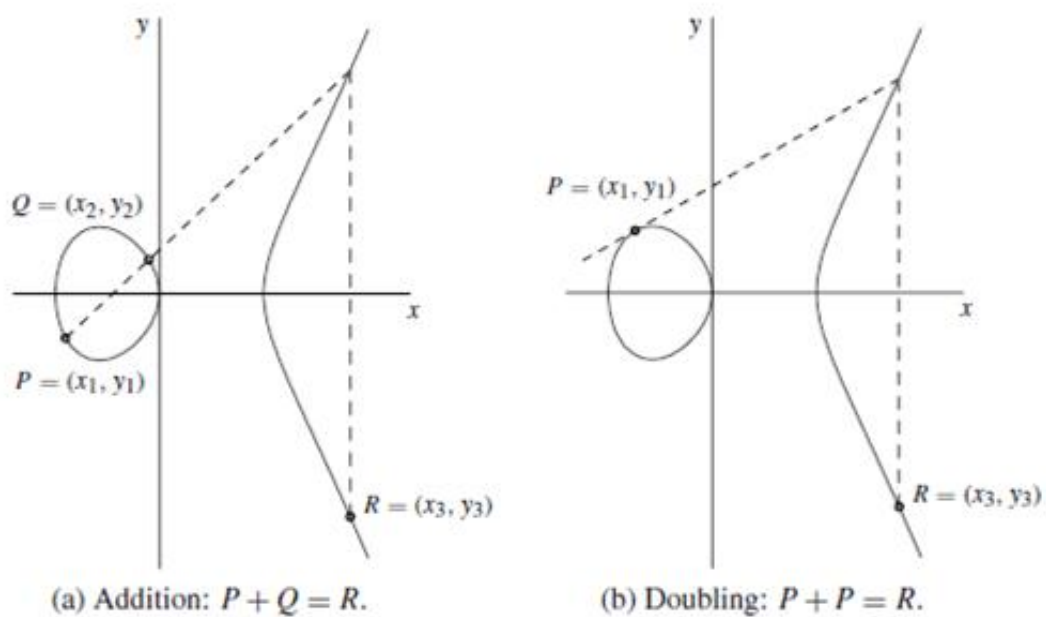


Figura 2: Soma em Curva Elíptica

2.5.4 Geração de chaves em curva elíptica

Seja E uma curva elíptica definida sobre o corpo finito F_p . Seja P um ponto em $E(F_p)$, e suponha que P tenha ordem prima n . Então o subgrupo cíclico de $E(F_p)$ gerado por P é

$$\langle P \rangle = \{ \infty, P, 2P, 3P, \dots, (n-1)P \}. \quad (2.3)$$

O primo p , a equação da curva elíptica E , e o ponto P e sua ordem n , são os parâmetros públicos do domínio. Uma chave privada é um inteiro d que é selecionado uniformemente de forma aleatória no intervalo $[1, n-1]$, e a chave pública correspondente é $Q = dP$. O problema de determinar d dados os parâmetros do domínio e Q é o problema do logaritmo discreto em curvas elípticas (ECDLP).

2.6 Criptografia baseada em identidades e sistemas isentos de certificados

O conceito de criptografia baseada em identidades (SHAMIR, 1984) procura reduzir a dificuldade causada pela necessidade de manutenção de uma infra-estrutura de chave pública, ou *PKI*. Nesse tipo de sistema a chave pública do usuário pode ser arbitrariamente escolhida, e uma autoridade de confiança gera sua chave privada. No entanto, a autoridade de confiança tem o conhecimento de todas as chaves privadas e poderia recuperar as informações de qualquer usuário.

Os sistemas de segurança isentos de certificados (ou, em uma definição mais precisa, auto-certificados) (AL-RIYAMI; PATERSON, 2003) foram concebidos com o objetivo de resolver o problema de comprometimento da chave de sistemas baseados em identidades. Nesses sistemas, a chave privada do usuário é composta por dois componentes: uma parcela baseada em identidades, e, desse modo, sujeita a comprometimento, e uma parcela convencional, porém não certificada. Esse tipo de sistema combina as melhores características de sistemas baseados em identidades e em certificados, porém continuam sendo de difícil especificação.

2.7 Emparelhamentos

Os mapeamentos bilineares, ou emparelhamentos (SAKAI; OHGISHI; KASAHARA, 2000; BONEH; FRANKLIN, 2001), tornaram possível o uso de criptografia baseada em identidades na prática. Emparelhamentos são formalmente definidos como se segue. Seja k um parâmetro de segurança e n um número primo de k bits. Sejam \mathbb{G}_1 , \mathbb{G}_2 e \mathbb{G}_T grupos de ordem n . Dizemos que $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ são grupos de emparelhamento se existe um mapeamento bilinear

$e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ que satisfaça às propriedades:

1. Bilinearidade: $\forall (S, T) \in \mathbb{G}_1 \times \mathbb{G}_2, \forall a, b \in \mathbb{Z}_n, e(aS, bT) = e(S, T)^{ab}$.
2. Não-degenerabilidade: $\forall S \in \mathbb{G}_1, e(S, T) = 1$ para todo $T \in \mathbb{G}_2$ se e somente se $S = O_{\mathbb{G}_1}$.
3. Computabilidade: $\forall (S, T) \in \mathbb{G}_1 \times \mathbb{G}_2, e(S, T)$ é eficientemente computável.

2.8 BLMQ

Nesta seção apresentamos características básicas e definições do BLMQ (BARRETO et al., 2005), um esquema de criptografia baseado em identidades. O esquema é composto pelos seguintes algoritmos:

- **Setup:** dado um parâmetro de segurança k , este algoritmo escolhe um número primo de k bits n , grupos de mapeamento bilinear $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ de ordem n que suportam um emparelhamento eficientemente computável e não degenerado $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$, geradores $P \in \mathbb{G}_1, Q \in \mathbb{G}_2$ e as funções de hash $h_0 : \mathbb{G}_T \times \{0, 1\}^* \rightarrow \mathbb{Z}_n^*, h_1 : \{0, 1\}^* \rightarrow \mathbb{Z}_n^*$. Uma chave mestra $s \xleftarrow{\$} \mathbb{Z}_n^*$ é também escolhida, para a qual a seguinte chave pública $P_{pub} = sP \in \mathbb{G}_1$ é associada. O gerador $g = e(P, Q) \in \mathbb{G}_T$ é também incluído entre os parâmetros públicos que são $params = (k, n, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, P, Q, g, P_{pub}, e, h_0, h_1)$.
- **Private-Key-Extract:** Toma como entrada o identificador $ID_A \in \{0, 1\}^*$ da entidade A e extrai a chave privada baseada em identidades $Q_A \leftarrow (h_1(ID_A) + s)^{-1} Q \in \mathbb{G}_2$ de A . A entidade A pode verificar a consistência dessa chave verificando que $e(h_1(ID_A)P + P_{pub}, Q_A) = g$. Esta configuração é chamada de estilo Sakai-Kasahara (SAKAI; KASAHARA, 2003).
- **Sign:** Para assinar $m \in \{0, 1\}^*$ sob a chave privada P_A , o assinante toma $u \xleftarrow{\$} \mathbb{Z}_n^*$ e calcula

1. $r \leftarrow g^u$
2. $h \leftarrow h_0(r, m)$
3. $S \leftarrow (u - h)Q_A$

A mensagem assinada é a tripla $(m, h, S) \in \{0, 1\}^* \times \mathbb{Z}_n^* \times \mathbb{G}_2$.

- **Verify:** dada um identidade ID_A , sob a recepção de (m, h, S) o verificador

1. $r \leftarrow e(h_1(ID_A)P + P_{pub})g^h$
2. $v \leftarrow h_0(r, m)$

O verificador aceita a mensagem assinada se e somente se $v = h$.

Pode-se mostrar que esse esquema é existencialmente infalsificável sob ataques de mensagens adaptativamente escolhidas (EUF-IBS-CMA abreviados) no modelo do oráculo aleatório sob a assunção do q -SDHP (BARRETO et al., 2005, section 3.1). Perceba que nesta descrição escolhemos definir $P_{pub} \in \mathbb{G}_1$, $Q_A \in \mathbb{G}_2$ para evitar a aritmética em \mathbb{G}_2 , mas uma descrição análoga com $Q_{pub} \in \mathbb{G}_2$, $P_A \in \mathbb{G}_1$ e mensagens assinadas em $\{0, 1\}^* \times \mathbb{Z}_n^* \times \mathbb{G}_1$ seria igualmente segura, enquanto mantemos a assinatura tão curta quanto possível na prática.

2.9 BDCPS

O protocolo de segurança BDCPS (BARRETO et al., 2008) integra as assinaturas baseadas em identidades do BLMQ, assinaturas de Schnorr (SCHNORR, 1991) e a cifrassinatura de Zheng (ZHENG, 1997) em um esquema isento de certificados conforme proposto por (AL-RIYAMI; PATERSON, 2003). Este protocolo foi criado especialmente para atender às necessidades deste projeto.

O protocolo é composto pelos 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 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 $ID_A \in \{0, 1\}^*$, o identificador de A e o valor público $y_A \in \mathbb{G}_T$, e calcula a chave privada baseada em identidade de

$A, Q_A \leftarrow (h_1(y_A, ID_A) + s)^{-1} Q \in \mathbb{G}_2$. A entidade A consegue verificar a consistência desta chave verificando se $e(h_1(y_A, ID_A)P + P_{pub}, Q_A) = g$. 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. $h_A \leftarrow h_0(r_A, y_A, ID_A)$
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 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. $r_A \leftarrow e(h_1(y_A, ID_A)P + P_{pub}, T_A)y_A^{h_A}$
2. $v_A \leftarrow h_0(r_A, y_A, ID_A)$

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 validade para a identidade ID_B e P_{pub} , e a chave privada do emissor $x_A \in \mathbb{Z}_n^*$, chave pública $y_A \in \mathbb{G}_T$ e a identidade 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. $h \leftarrow h_3(r, m, y_A, ID_A, y_B, ID_B)$
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 corpos), 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 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 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. $v \leftarrow h_3(r, m, y_A, ID_A, y_B, ID_B)$

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.

3 DISCUSSÃO

3.1 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 serviços básicos de segurança, a saber, confidencialidade, integridade e autenticidade às mensagens SMS, permitindo ao usuário assinar, cifrar, decifrar e verificar mensagens enviadas pela rede GSM. As soluções adotadas no projeto envolvem criptografia de chave pública, criptografia baseada em identidade e também esquemas auto-certificados, conforme definidos no capítulo anterior. A adoção dessa combinação deve dispensar a existência de um diretório de chaves públicas, uma vez que o uso de certificados convencionais exigiria uma infra-estrutura e demandaria um consumo de banda impraticáveis em uma rede de telefonia celular com SMS.

O software será aplicável em áreas que requeiram segurança da informação 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 (ISO, 1994), sendo transparente à arquitetura interna da rede

GSM. Essas duas qualidades tornam possível sua viabilidade em diversos cenários.

O *Sistema de SMS Seguro* não é responsável por garantir a confiabilidade e disponibilidade de entrega das mensagens. Essa função é de responsabilidade do fornecedor do serviço móvel. O sistema também não garante proteção quanto à clonagem do telefone, mas pode garantir a autenticidade do emissor e indiretamente detectar a clonagem caso o usuário já tenha se cadastrado na autoridade de confiança.

O sistema também não oferece o serviço de irretratabilidade, uma vez que a chave privada do receptor é utilizada no processo de verificação de autenticidade da mensagem. Desse modo, um usuário *B* não pode convencer um terceiro de que uma mensagem foi enviada por *A* sem comprometer sua chave privada.

3.2 Métricas e Restrições

A seguir, definimos as métricas e restrições do sistema.

- 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 no número de bytes ocupados por dados de controle do algoritmo. Estabelecemos que este *overhead* não deve ultrapassar 25% do espaço total da mensagem.
- 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 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.
- Celular desbloqueado: Para que o sistema execute no ambiente do celular ele deve estar desbloqueado para execução de aplicativos Java.
- Interface de troca de dados: o celular deverá possuir alguma interface para poder fazer o *download* do aplicativo.

Um certificado digital típico ocupa entre 2KB e 4KB, e 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. 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. 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, inicialmente, ser capaz de atender aos requisitos do sistema.

4 ANÁLISE DE REQUISITOS DO SISTEMA

4.1 Requisitos funcionais

Após a análise do escopo e das restrições do sistema, foram levantados os seguintes requisitos funcionais:

4.1.1 Cifrassinatura de mensagem

Introdução/Propósito O sistema deverá fornecer serviço de confidencialidade através da funcionalidade de cifrassinatura das mensagens SMS, ou seja, as mensagens, para trafegar na rede GSM deverão estar encriptadas e assinadas.

Estímulo/Resposta O estímulo neste caso provém do usuário ao requisitar o serviço de cifrassinatura e a resposta do sistema é realizar o serviço enviando a mensagem cifrassinada.

4.1.2 Verificação de mensagem

Introdução/Propósito O sistema deverá fornecer ao usuário o serviço de ele poder verificar o autor de uma mensagem recebida como o de ler seu conteúdo.

Estímulo/Resposta O usuário requisita ao sistema a verificação de uma determinada mensagem e o sistema processa a mensagem devolvendo a ve-

racidade de sua validade e seu conteúdo.

4.1.3 Envio de mensagem

Introdução/Propósito: O sistema deve fornecer ao usuário o serviço de envio de uma mensagem para o destinatário desejado.

Estímulo/Resposta: Usuário deseja enviar uma mensagem SMS criptografada e requisita o serviço ao sistema que efetua o envio.

4.1.4 Geração de chave privada

Introdução/Propósito: O sistema deve fornecer para um novo usuário o serviço gerar sua chave privada e entregá-lo de forma segura. O Key Generation Bureau possui essa responsabilidade.

Estímulo/Resposta: Novo usuário requisita ao sistema a geração de sua chave privada. A requisição chega até o KGB que gera a chave privada do usuário e a retorna por meio de uma mensagem segura.

4.2 Requisitos não-funcionais

4.2.1 Usabilidade

O tamanho da chave privada não deve prejudicar a usabilidade do software. O método de entrada das mensagens deve ser semelhante ao dos aparelhos celulares convencionais.

4.2.2 Desempenho

A aplicação deve ser capaz de cifrar ou decifrar uma mensagem em menos de 5 segundos.

4.2.3 Confiabilidade

O software deverá apresentar *MTTF* de 1 ano. Entende-se como falha a parada do software pela subida de uma exceção não tratada. Essa medida ignora falhas de componentes externos ao software (hardware do celular, plataforma Java).

4.2.4 Disponibilidade

O software deverá apresentar disponibilidade de 99%. Entende-se como disponibilidade a razão entre as tentativas bem sucedidas de acessar o software e o total de tentativas. Ou seja, a cada 100 tentativas de acessar o software apenas uma não terá sucesso.

4.2.5 Compatibilidade

O software deverá ser compatível com todos os dispositivos celulares equipados com a plataforma Java ME (Micro Edition), desde que estejam desbloqueados, e com a configuração CLDC.

4.2.6 Portabilidade

O software deverá ser portátil para a plataforma Java SE (Standard Edition) tendo em vista o uso da aplicação como interface com web services.

4.2.7 Segurança

O software deve garantir que o destinatário da mensagem e apenas ele, além do remetente, tenha acesso ao seu conteúdo em tempo viável. O software também deve garantir a integridade da mensagem em relação a corrupções maliciosas ou acidentais durante o tráfego.

4.3 Casos de uso

O sistema possui 3 atores, a saber

1. **Usuário:** o usuário do sistema móvel pessoal que enviará e receberá mensagens através do canal seguro.
2. **KGB:** é a autoridade de segurança, responsável por gerar a parcela da chave privada baseada em identidades dos usuários.
3. **Sistema:** o sistema deverá acionar-se sempre que uma mensagem SMS chegar ao *listener* da porta especificada.

Nas seções abaixo detalharemos cada caso de uso do sistema, representados sucintamente no diagrama da Figura 3.

4.3.1 Cadastrar-se no sistema

1. **Descrição:** Novo usuário deseja usar o sistema pela primeira vez e precisa efetuar as configurações necessárias para poder receber sua chave privada.
2. **Evento Iniciador:** Usuário seleciona a opção de primeiro uso do sistema
3. **Atores:** Usuário, KGB.

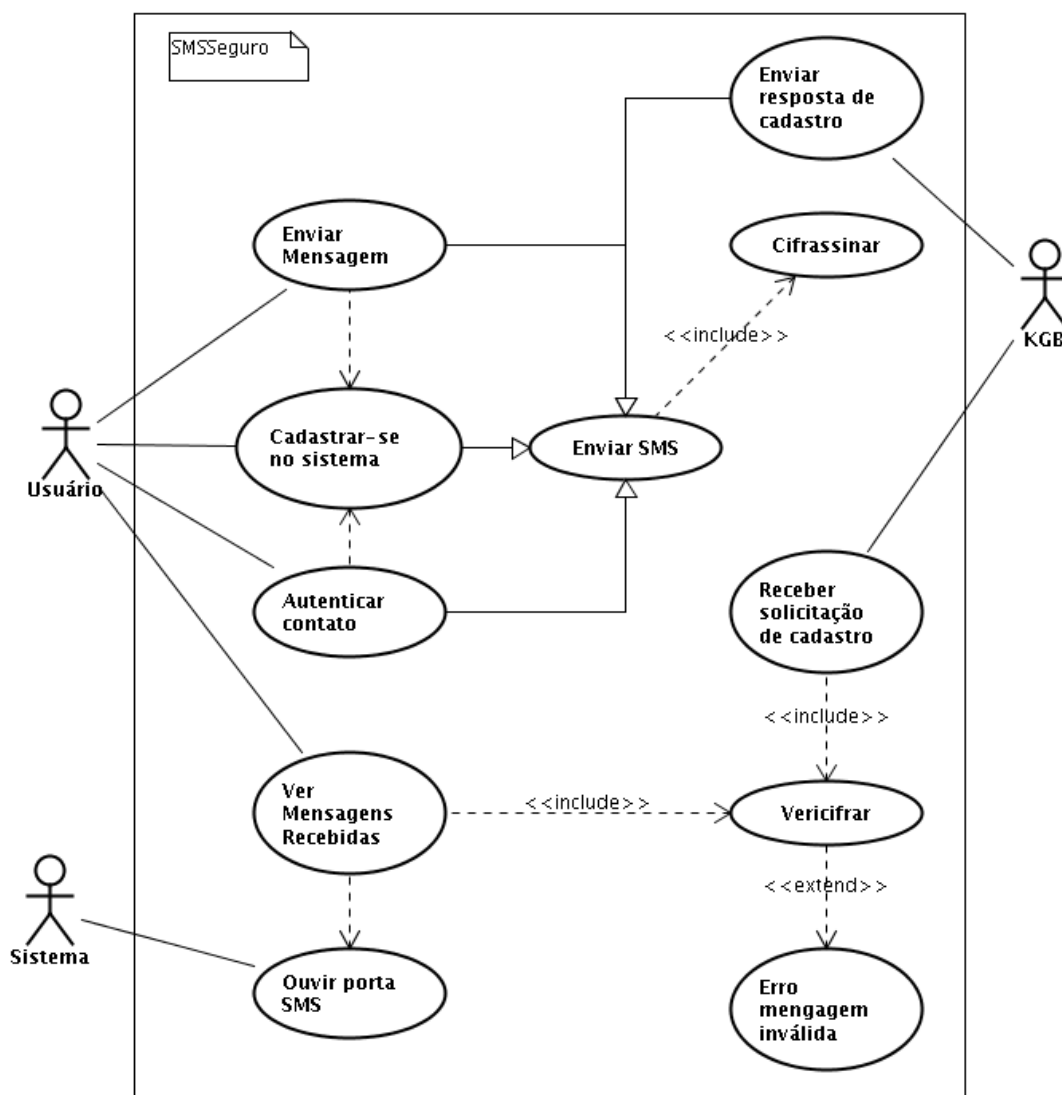


Figura 3: Diagrama de casos de uso do sistema

4. **Pré-condição:** Sistema de SMS seguro apresentando sua tela inicial.

5. **Seqüência de Eventos:**

- (a) Usuário seleciona o botão de primeiro uso.
- (b) Sistema pede a entrada de uma nova senha privada do usuário.
- (c) Usuário cadastra uma nova senha no sistema e confirma.
- (d) Sistema exibe notificação de envio de mensagem de controle para a KGB.
- (e) Usuário confirma o envio e sistema transmite a mensagem.

- (f) Ao receber a mensagem, a KGB gera a chave privada do usuário e retorna uma mensagem segura contendo a chave gerada.
 - (g) O sistema do usuário recebe a mensagem da KGB contendo sua chave privada.
 - (h) Sistema pede novamente a senha do usuário e extrai a chave privada do usuário
 - (i) O sistema verifica se a chave privada recebida é válida.
 - (j) O sistema armazena a nova chave no celular.
6. **Pós-Condição:** Sistema volta para a tela inicial e usuário está apto a autenticar novos contatos para trocar mensagens.

7. **Extensões:**

- (a) Caso haja algum problema na geração ou na mensagem que contém a chave privada do usuário ou ainda se outra entidade tentou se passar por KGB então o sistema exibe mensagem de chave inválida ao usuário.

4.3.2 Autenticar novo contato

1. **Descrição:** Quando um usuário quiser trocar mensagem com um contato que ainda não foi autenticado pelo Sistema de SMS Seguro deverá requisitar sua autenticação.
2. **Evento Iniciador:** Usuário deseja autenticar um novo contato para enviá-lo uma mensagem.
3. **Atores:** Usuário que quer se comunicar, usuário recebedor da mensagem.
4. **Pré-condição:** Sistema exibe tela inicial.

5. Seqüência de Eventos:

- (a) Usuário seleciona a opção de autenticar novo contato.
- (b) Usuário insere o número do telefone do novo contato e seleciona OK.
- (c) Sistema exibe notificação de envio de SMS para o contato informado.
- (d) Usuário confirma o envio da mensagem SMS.
- (e) Sistema envia requisição de autenticação para o novo contato.

6. **Pós-Condição:** Sistema volta para a tela inicial.

7. Extensões:

- (a) Sistema exibe notificação de erro no envio da mensagem caso o serviço de envio esteja indisponível. (Passo 5e).

4.3.3 Enviar Mensagem

- 1. **Descrição:** Usuário deseja compor e enviar uma nova mensagem SMS para outro usuário.
- 2. **Evento Iniciador:** Usuário seleciona botão de envio de nova mensagem.
- 3. **Atores:** Usuário emissor da mensagem.
- 4. **Pré-condição:** Sistema de SMS seguro apresentando sua tela inicial.
- 5. **Seqüência de Eventos:**
 - (a) Usuário emissor da mensagem seleciona botão de envio de mensagem.

- (b) Sistema exige que o usuário indique o destinatário da mensagem.
- (c) Usuário seleciona o destino da lista de contatos exibida.
- (d) Usuário compõe a mensagem a ser enviada
- (e) Usuário confirma o envio da mensagem para o destinatário escolhido.
- (f) Sistema exibe notificação de envio da mensagem.

6. **Pós-Condição:** A notificação de mensagem enviada é exibida e o sistema retorna à tela inicial.

7. **Extensões:**

- (a) Sistema exibe notificação de erro no envio da mensagem caso o serviço de envio esteja indisponível. (Passo 5e).

8. **Inclusões:**

- (a) Sistema busca todos os contatos da lista de contatos do celular para exibí-los.
- (b) Caso de uso 4.3.5.

4.3.4 Recepção de Mensagem

1. **Descrição:** Quando uma nova mensagem chega no celular o sistema deve captá-la e fazer seu tratamento.
2. **Evento Iniciador:** Chega uma nova mensagem do Sistema de SMS Seguro no celular de um usuário.
3. **Atores:** Sistema operacional.
4. **Pré-condição:** Celular do usuário ligado.

5. Seqüência de Eventos:

- (a) Celular recebe a nova mensagem e a coloca na fila.
- (b) Sistema operacional do celular capta a mensagem e requisita ao usuário que inicialize a aplicação caso ela não esteja em execução.
- (c) O sistema identifica a primitiva da mensagem e trata de forma correspondente.

6. **Pós-Condição:** a mensagem está processada e o sistema está exibindo a tela inicial.

7. Extensões:

- (a) Sistema trata mensagem cifrada e assinada. (Passo 5c)
- (b) Sistema trata mensagem de autenticação de usuário.(Passo 5c)
- (c) Sistema trata mensagem de pedido da chave privada. (Passo 5c)
- (d) Sistema trata mensagem de entrega de chave privada. (Passo 5c)

4.3.5 Encriptar/Assinar Mensagem

1. **Descrição:** Usuário escreveu uma mensagem para alguém e deseja cifrá-la e assiná-la.
2. **Evento Iniciador:** Usuário requisita envio de mensagem cifrada e assinada ao sistema.
3. **Atores:** Usuário que deseja enviar uma mensagem segura.
4. **Pré-condição:** Usuário está com a mensagem pronta na tela de envio.
5. **Seqüência de Eventos:**
 - (a) Usuário seleciona a opção de enviar a mensagem.

(b) Sistema cifra e assina a mensagem e exibe tela de confirmação de envio.

(c) Usuário confirma o envio e sistema transmite a mensagem segura.

6. **Pós-Condição:** Sistema volta para a tela inicial.

4.3.6 Decriptar/Verificar Mensagem

1. **Descrição:** Usuário deseja visualizar uma mensagem na sua caixa de entrada.

2. **Evento Iniciador:** Usuário abre a caixa de entrada do sistema.

3. **Atores:** Usuário.

4. **Pré-condição:** Sistema de SMS seguro apresentando mensagens recebidas na caixa de entrada. Sequência de Eventos:

(a) Usuário escolhe a mensagem que deseja visualizar e seleciona ok.

(b) Sistema verifica e decifra a mensagem.

(c) Sistema exibe a mensagem clara para que o usuário possa lê-la.

5. **Pós-Condição:** Sistema exibindo mensagem clara para o usuário.

5 ESCOLHA DO ESQUEMA CRIPTOGRÁFICO

5.1 Por que criptografia em curvas elípticas?

Há vários critérios que precisam ser considerados ao selecionar-se uma família de esquemas de chave pública para uma determinada aplicação.

Os princípios são:

- Funcionalidade. A família de chave pública fornece as habilidades desejadas?
- Segurança. O que garante que os protocolos são seguros?
- Eficiência. Para o nível de segurança desejado, os protocolos fornecem os objetivos de eficiência.

Outros fatores que podem influenciar uma decisão incluem a existência de padrões de melhores práticas desenvolvidos por organizações de padronização confiáveis, a disponibilidade de produtos criptográficos comerciais, coberturas de patentes, e extensão das aplicações existentes.

As famílias do RSA, do logaritmo discreto e das curvas elípticas introduziram na criptografia de chave pública todas as funcionalidades básicas esperadas - encriptação, assinaturas e troca de chaves.

Durante os anos, pesquisadores desenvolveram técnicas para modelar e provar a segurança dos protocolos RSA, logaritmo discreto e curvas elípticas

sob hipóteses razoáveis. A questão fundamental da segurança que permanece é a dificuldade do problema matemático subjacente que é necessário para a segurança de todos os protocolos em uma família de chave pública - o problema da fatoração inteira para sistemas RSA, o problema do logaritmo discreto para os sistemas baseados em logaritmos discretos e o problema de logaritmo discreto em curvas elípticas em sistemas baseados em curvas elípticas. A dificuldade percebida desses problemas impacta diretamente na eficiência uma vez que ela dita os tamanhos do domínio e dos parâmetros das chaves. Isso, por outro lado, afeta a eficiência das operações aritméticas subjacentes.

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 consideravelmente o tamanho das chaves, uma vez que a relação entre o tamanho das chaves e o nível de segurança é exponencial, a chave cresce muito rapidamente quando aumenta-se o nível de segurança desejado.(MENEZES; OORSCHOT; VANSTONE, 1996)

Em paralelo, um aumento equivalente no nível de segurança de criptografia em curvas elípticas acarreta menor aumento no tamanho das chaves. Este fato ocorre devido à relação entre o nível de segurança de criptografia em curvas elípticas e o tamanho das chaves, que é uma relação diretamente proporcional (cresce linearmente). Para uma chave k , tem-se um nível de segurança de aproximadamente $k/2$. (MENEZES; OORSCHOT; VANSTONE, 1996)

5.2 BLMQ

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

de criptografia baseada em identidades.

O esquema foi escolhido por, aparentemente, atender aos requisitos estabelecidos. 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, pela primeira vez, o uso desse tipo de criptografia viável em ambientes móveis como a telefonia celular. Além disso, o uso de uma assinatura de 160 bits garantiria um nível de segurança equivalente ao do RSA de 1024 bits (KALISKI, 2003), maximizando o espaço útil da mensagem.

5.2.1 Testes de viabilidade

O esquema foi parcialmente implementado em 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 fornecessem a classe *BigInteger*. Algumas adaptações no esquema em si foram feitas, como inversão da ordem das curvas utilizadas, porém sem efeitos consideráveis.

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

Essa implementação inicial demandava muito tempo computacional para inicializar as classes e realizar as operações de emparelhamento. Emparelhamentos são extensamente utilizados pelo BLMQ, como nos algoritmos de *signcrypt* e *unsigncrypt*, o que impunha grande *overhead* às operações do sistema.

Como estes tempos não atendiam às métricas e aos requisitos de usabilidade do projeto, fez-se necessário buscar outras soluções. Estas dificuldades serviram como motivação para a criação de um esquema inovador. Como resultado de pesquisas realizadas, foi idealizado o protocolo, brevemente descrito a seguir.

5.3 BDCPS

O esquema proposto por (BARRETO et al., 2008) 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 convencionais para validar sua chave pública.

5.3.1 Vantagens do esquema

Dentre as operações realizadas nos diversos algoritmos, a que apresenta maior custo computacional é a operação de emparelhamento. Observa-se que os algoritmos de *Signcrypt* e *Unsigncrypt* não executam nenhum emparelhamento. Estes são os algoritmos que serão usados mais vezes, já que são usados toda vez que deseja-se enviar ou ler uma mensagem cifrassinada. Os emparelhamentos são executados apenas nos algoritmos de validação e veri-

ficação das chaves públicas. No entanto, seu custo é amortizado pois esses algoritmos são executados apenas uma vez para cada canal seguro estabelecido para um par de usuários, isto é, apenas na primeira interação.

Estabelece-se, aqui, um esquema de assinaturas auto-certificado, isto é, não é necessária a interação com uma autoridade de confiança para que um par de usuários estabeleça um canal seguro. A interação com a autoridade de confiança é necessária apenas quando um usuário gera o seu par de chaves.

5.3.2 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 estavam de acordo com as métricas estabelecidas e bem mais eficientes em relação ao esquema inicialmente estudado.

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 preliminares são apresentados nas tabelas 2 e 3. Foram feitos testes de viabilidade 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	Nokia 6275 (s)	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

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

Operação	Nokia 6275 (s)	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

5.4 Análise dos resultados preliminares

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 do BLMQ e, como observado em (BARRETO et al., 2008), menores do que os de outros protocolos, como o RSA, para um mesmo nível de segurança.

Desse modo, por ter sido especificamente projetado para as necessidades do cenário e ter apresentado ótima eficiência, o BDCPS foi escolhido como protocolo de segurança para nosso sistema.

6 ESPECIFICAÇÃO E PROJETO

6.1 Arquitetura

O sistema será dividido em dois módulos: o módulo do cliente, que possibilitará ao usuário a troca de mensagens com outros usuários, e o módulo da autoridade certificadora, ou *KGB*, que será responsável pela geração de parte da chave privada dos usuários, conforme a Figura 4.

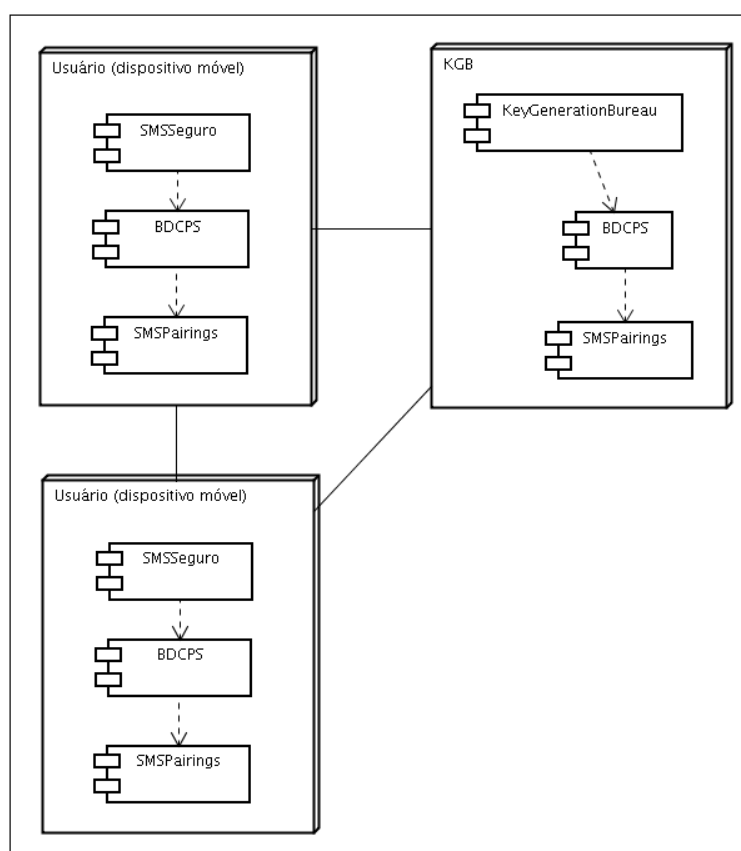


Figura 4: Diagrama de implantação do sistema

6.2 Classes

6.2.1 Descrição

As classes do sistema são divididas em 4 pacotes:

- **Protocol:** pacote que contém as classes que implementam o protocolo de segurança BDCPS.
- **Data:** pacote que contém as classes que gerenciam a persistência dos dados da aplicação.
- **Application:** pacote que contém a interface gráfica e as classes que gerenciam os serviços do usuário e da autoridade de segurança (Key-GenerationBureau).
- **Messaging:** pacote que contém as classes que gerenciam a interface com o serviço de mensagens SMS e a serialização dos dados trafegados nas mensagens binárias.

A figura 5 apresenta um diagrama de classes simplificado do sistema. O apêndice B apresenta diagramas de classe detalhados de cada pacote.

6.3 Especificação do protocolo de troca de mensagens

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

¹Os bytes de uma mensagem serão numerados iniciando de 1.

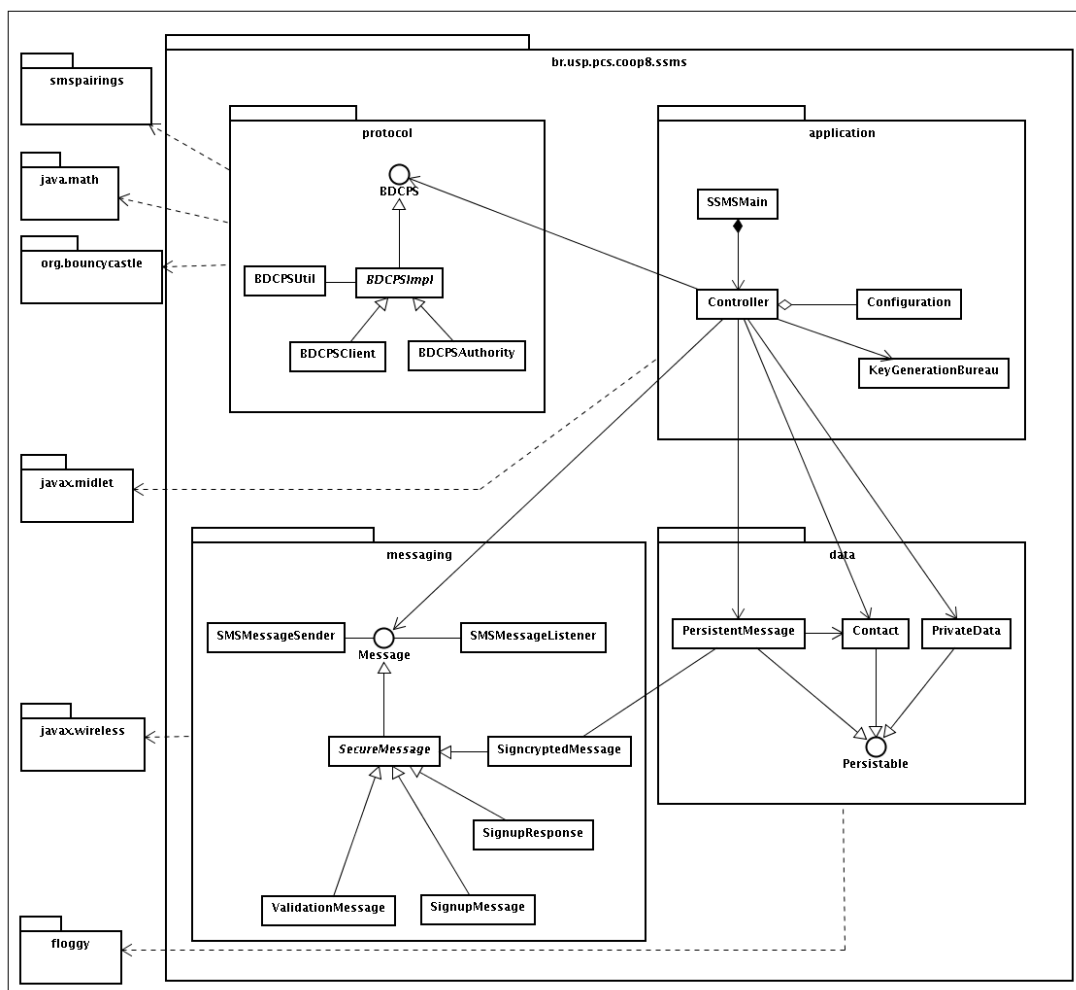


Figura 5: Diagrama de classes do sistema

A figura 6 ilustra o fluxo de estabelecimento da chave privada de um usuário com a autoridade certificadora. O figura 7 ilustra o fluxo de autenticação e troca de mensagens entre dois usuários. As seções seguintes especificam o formato de cada primitiva do sistema.

6.3.1 SignupMessage

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

- Byte 1: Byte fixo que identifica uma mensagem do nosso protocolo. Valor $0x42$.

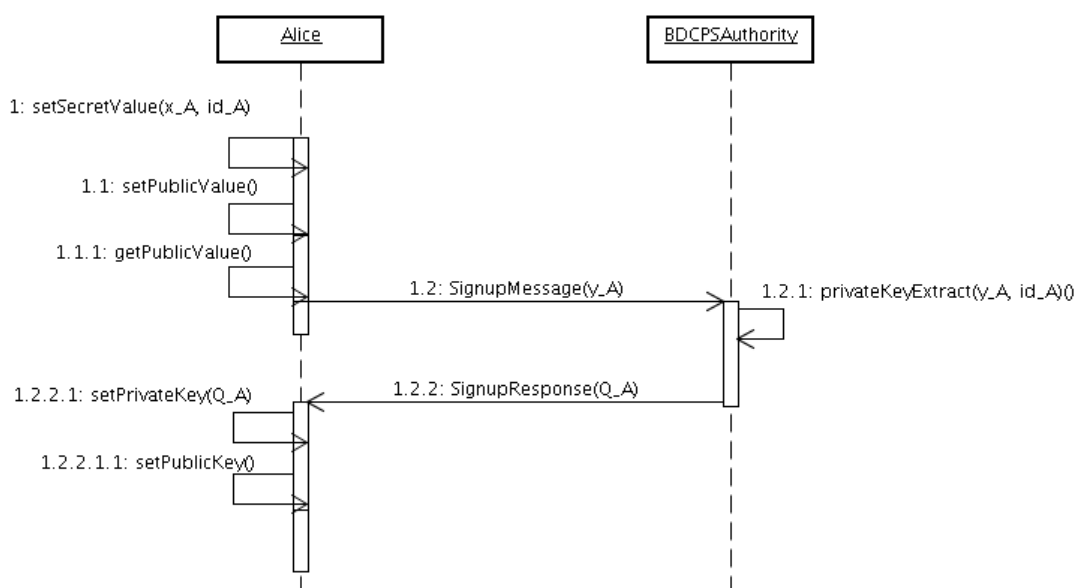


Figura 6: Estabelecimento da chave privada com a autoridade certificadora

- Byte 2: Byte fixo, identifica a primitiva **SignupMessage**. 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 .

6.3.2 SignupResponse

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

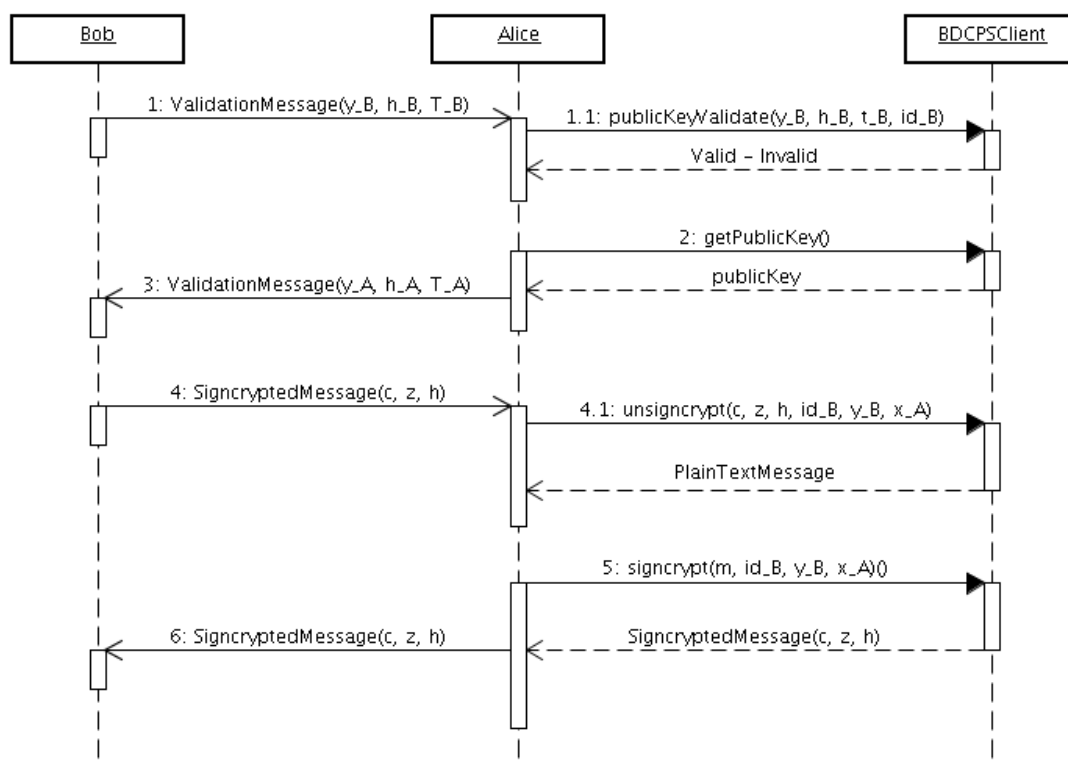


Figura 7: Fluxo de comunicação entre dois usuários

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

Figura 8: SignupMessage

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 0x42.
- Byte 2: Byte fixo, identifica a primitiva **SignupResponse**. 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	Param. 1	Param. 2	Param. 3
0x50	0x01	K	0x00	Tam. c	Tam. h	Tam. z	... c h z ...

Figura 9: SignupResponse

6.3.3 ValidationMessage

Representa a mensagem pela qual 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 0x42.
- Byte 2: Byte fixo, identifica a primitiva **ValidationMessage**. 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 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 .

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Param. 1	Param. 2	Param. 3
0x50	K	0x00	0x00	Tam. y_A	Tam. h_A	Tam. T_A	... y_A h_A T_A ...

Figura 10: ValidationMessage

6.3.4 SigncryptedMessage

Representa uma mensagem cifrassinada a ser trocada entre usuários.

- Byte 1: Byte fixo que identifica uma mensagem do nosso protocolo. Valor $0x42$.
- Byte 2: Byte fixo, identifica a primitiva **SigncryptedMessage**. Valor $0x03$.
- 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	Param. 1	Param. 2	Param. 3
0x50	0x03	K	0x00	Tam. c	Tam. h	Tam. z	... c h z ...

Figura 11: SigncryptedMessage

7 IMPLEMENTAÇÃO

A implementação foi feita na linguagem *Java*, plataforma *J2ME MIDP-1.0 CLDC-1.1*.

7.1 Ambiente de desenvolvimento

Para o desenvolvimento do código, foi usado o *IDE Netbeans 6.0.1*, integrado com o *Sun Java (TM) Wireless Toolkit 2.5.2 for CLDC (WTK)*. Usamos o emulador do *WTK* para auxiliar o processo de desenvolvimento. Também foi usado o *Subversion*¹ para controle de versão e coordenação do trabalho em equipe, hospedado nos servidores do *Google Code*².

7.2 Bibliotecas utilizadas

7.2.1 SMSPairings

A biblioteca *SMSPairings* foi fornecida por nosso orientador. Contém classes que implementam curvas elípticas e emparelhamentos bilineares, de uma forma otimizada para a ordem de grandeza de nossas chaves.

¹sistema de gerenciamento de configuração disponível em <http://subversion.tigris.org>

²<http://code.google.com/>

7.2.2 BouncyCastle

O *BouncyCastle* é uma biblioteca que contém implementações de vários algoritmos de criptografia e *hashs*. Existe uma versão da biblioteca implementada para o ambiente *J2ME*, a qual usamos. Fez-se necessário o uso desta biblioteca devido à ausência dos pacotes *java.security* e *javax.crypto* no ambiente dos celulares usados. Estas bibliotecas são opcionais do *framework J2ME*, e não estava presente nos celulares que usamos para desenvolver e testar. Assim, usamos a biblioteca *BouncyCastle* como substituta para estes pacotes ausentes.

7.2.3 Floggy

O *Floggy* é um *framework* de persistência de dados em ambiente *J2ME*, desenvolvido no Brasil. Foi essencial para nosso projeto para persistir objetos como mensagens cifradas recebidas, contatos validados e dados do protocolo, como a chave privada parcial Q_A ;

7.3 Escolha de parâmetros

7.3.1 Escolha do tamanho de chave

Nossa aplicação é capaz de trabalhar com um tamanho de chave de k bits, se $k \in \{80, 96, 104, 112, 117, 127, 142, 160, 176, 187, 256, 272, 313\}$. Não podemos simplesmente usar qualquer valor, pois é necessário o uso de uma curva adequada (MNT4) para cada valor. Nem sempre é possível encontrar uma curva adequada para um dado tamanho em bits, assim ficamos limitados a usar valores que têm curvas conhecidas associadas a ele. Nota-se um grande intervalo entre os valores 187 e 256, pois não foi possível encontrar curvas adequadas

no interior deste intervalo.

Na escolha do valor ideal, é preciso considerar o tamanho que os parâmetros irão consumir no espaço útil da mensagem, além do nível de segurança.

Foi escolhido o número 176 como tamanho de chave padrão³, pois este fornece um nível de segurança equivalente ao do RSA com 704 bits, o que já representa um nível de segurança adequado para nossa aplicação. Por outro lado, não é um valor tão grande que chega a ocupar muitos bytes da mensagem, cada parâmetro enviado terá 176 bits (22 bytes), ocupando apenas 15,7% do tamanho de um segmento SMS (que comporta no máximo 140 bytes).

7.3.2 Escolha da porta SMS

Quando se envia um SMS, associa-se a ele uma porta. A porta é um valor inteiro entre 0 e 65535 que serve para que o receptor encaminhe a mensagem recebida a uma aplicação específica. No caso de um SMS de texto normal, o valor da porta é 0. Quando um sistema operacional de um telefone móvel recebe uma mensagem com o valor de porta 0 ele aciona as rotinas do próprio sistema operacional para tratá-la, como armazená-la na caixa de entrada e tocar um som de alerta para o usuário. No caso de a porta ser diferente de 0, o sistema operacional procura numa área chamada *PushRegistry* por algum aplicativo instalado que deseja receber mensagens nesta porta (é como se o aplicativo estivesse escutando a porta) e assim executa aplicativo registrado, que irá tratar a mensagem recebida⁴.

Sendo assim, escolhemos arbitrariamente o valor 50001 para usar como

³O valor pode ser facilmente alterado para qualquer um dos valores suportados sem a necessidade de grande alteração no código do sistema

⁴Em alguns celulares, o sistema operacional pode pedir uma confirmação do usuário antes de executar a aplicação automaticamente.

a porta de nossa aplicação. A aplicação envia e se registra para escutar mensagens nesta porta.

7.4 Telas do sistema

As telas do sistema são apresentadas no apêndice C.

8 RESULTADOS

Após implementado, o sistema foi testado usando diferentes celulares e diferentes operadoras.

8.1 Desempenho

Foram executados testes de desempenho com a versão final do sistema. Na tabela 4 podemos observar os tempos das operações usando chaves de 176 bits.

No apêndice D apresentamos um gráfico demonstrando como o tempo da operação de cifrassinatura varia conforme o tamanho da mensagem. Também acrescentamos gráficos demonstrando como estes tempos variam de acordo com o tamanho de chave usado. Neste teste, foram utilizados os valores de chave suportados pela aplicação, explicitados na seção 7.3.1

Tabela 4: Testes com a implementação final (chaves de 176 bits)

Operação	Nokia E51(ms)	Nokia 6275(ms)	Emulador(ms)
Set-Public-Value	66,9	750,6	204,5
Private-Key-Extract	379,0	4381,7	1033,9
Check-Private-Key	1164,9	12171,1	3209,9
Set-Public-Key	379,5	4332,4	1013,3
Public-Key-Validate	1192,6	13112,0	3455,8
Signcryption	302,4	1633,5	428,8
Unsigncryption	266,7	1957,0	492,2

O apêndice D apresenta gráficos e tabelas com dados de desempenho

mais detalhados.

8.2 Testes entre operadoras

Realizamos testes de envio de mensagens de nossa aplicação entre celulares habilitados para operadoras distintas. Foram testadas as principais operadoras do estado de São Paulo: *Vivo*, *Tim* e *Claro*. Observou-se que em alguns casos o SMS enviado não era recebido na porta *SMS* especificada no envio: mesmo sendo encaminhada à porta 50001, a mensagem era recebida na porta 0 (a porta padrão do SMS), não alcançando a aplicação e sim a caixa de entrada padrão do celular.

Na tabela 5 apresentamos o resultado de nossos testes, informando quais são as combinações em que o nosso sistema funcionou corretamente.

Tabela 5: Compatibilidade entre operadoras

	Vivo	Tim	Claro
Vivo	OK	NOK	NOK
Tim	NOK	OK	OK
Claro	NOK	OK	OK

Este problema ocorre devido às implementações internas das integrações entre as operadoras, sendo impossível resolvê-lo em nível de aplicação. Seria necessário uma negociação com as operadoras, solicitando a completa integração de uma porta específica para o uso de nosso sistema. Por estar fora do escopo de nosso projeto, este problema não foi tratado.

9 CONCLUSÃO

Ao longo do trabalho, foi possível observar que a criptografia em curvas elípticas (*ECC*) já é uma alternativa viável em ambientes com restrições, principalmente de banda, como os serviços de telefonia móvel, apesar de sua considerável exigência de poder computacional. No entanto, o uso de esquemas híbridos, como o aqui proposto, alia as melhores qualidades dos dois paradigmas, possibilitando a implantação imediata de sistemas de segurança completos no contexto de dispositivos e aplicações móveis. Como resultado deste trabalho, obteve-se um esquema eficiente de criptografia de chave pública auto-certificado, baseado em identidades, onde os próprios usuários podem se autenticar na rede sem a necessidade de um diretório de chaves públicas, que era utilizado em criptografia de chave pública.

9.1 Análise dos resultados

A partir das implementações e testes dos algoritmos BLMQ e BDCPS, pudemos constatar resultados importantes que nos levaram a desistir da primeira solução e desenvolver a segunda. A seguir são enumerados resultados importantes que justificaram a escolha e desistência da primeira solução e a necessidade do desenvolvimento da segunda:

O BLMQ oferecia vantagens como a dispensa de um diretório *online* de chaves públicas, chaves de tamanho reduzido, no caso 160 bits, e a facili-

dade de validação de chaves públicas. Porém não se mostrou eficiente em aparelhos celular. As metas que não obtiveram sucesso são listadas a seguir:

- Tempo de inicialização: O tempo para inicialização das curvas e geração dos parâmetros no celular foi da ordem de 2 minutos deveria ser aguardado quando a aplicação fosse utilizada pela primeira vez.
- Tempo de emparelhamento: As implementações de emparelhamento adotadas ETA e ATE gastaram, cada uma, tempos próximos a 4 segundos no celular. Devido ao fato do método de verificação utilizar duas vezes a custosa função de emparelhamento, tínhamos tempo de verificação de pelo menos 8 segundos, desviando a solução da meta estabelecida para verificação de uma mensagem.

Já os resultados obtidos a partir dos testes com o BDCPS foram positivos em relação às metas. Dois celulares foram testados, sendo que um deles, o *Nokia 6275*, tem um processador 12 vezes mais lento que o outro, o *Nokia E51*. Em ambos os testes, as metas de tempo de assinatura e de verificação foram obedecidas com tempos gastos menores que 2 segundos para as duas principais operações. Além disso, as chaves também possuíam tamanho reduzido, de 176 bits, obedecendo à meta de ocupação de espaço útil da mensagem.

9.2 Perspectivas futuras

Apesar de já termos apresentado um sistema de segurança funcional, pronto para o uso, existem ainda pontos em que ele pode ser aperfeiçoado. Especificamente, o protocolo de segurança utilizado não fornece os serviços de irretratabilidade e revogação de chaves. O serviço de irretratabilidade poderia ser implantado com o uso de outros protocolos de cifrassinatura, no

entanto ao custo de degradação do desempenho devido ao uso mais extenso de emparelhamentos (BARRETO et al., 2008, section 3).

Por outro lado, são amplas as possibilidades de aplicações do sistema em ambientes reais, entre as quais podemos citar transações bancárias, pagamentos por celular e comunicação corporativa. Acreditamos que conceitualmente o sistema já esteja maduro o suficiente para este tipo de aplicação, ficando pendentes apenas estudos mais detalhados sobre as necessidades específicas de cada cenário de implantação.

9.3 Considerações finais

No decorrer do desenvolvimento deste trabalho, superamos as dificuldades tecnológicas impostas de maneira inovadora, gerando pesquisa e produzindo publicações (CRUZ et al., 2008; BARRETO et al., 2008) reconhecidas no meio acadêmico nacional. Esperamos, também, ter aberto caminho para novas pesquisas e desenvolvimentos na área de criptografia para dispositivos móveis, ainda pouco explorada.

REFERÊNCIAS

- AL-RIYAMI, S. S.; PATERSON, K. G. Certificateless public key cryptography. In: *Advanced in Cryptology – Asiacrypt'2003*. [S.l.]: Springer, 2003. (Lecture Notes in Computer Science, v. 2894), p. 452–473.
- BARRETO, P. S. L. M. et al. Toward efficient certificateless signcryption from (and without) bilinear pairings. In: *Anais do VIII Simpósio Brasileiro em Segurança da Informação de Sistemas Computacionais*. Gramado, RS, Brasil: Sociedade Brasileira de Computação, 2008. p. 115–125.
- _____. 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.
- CRUZ, E. et al. *Construção de um Sistema de SMS Seguro*. Gramado, RS, Brasil: Sociedade Brasileira de Computação, 2008. 413–421 p.
- DIFFIE, W.; HELLMAN, M. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-31, n. 22, p. 644–654, 1976.
- ELGAMAL, T. A public-key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, IT-31, n. 4, p. 469–472, 1985.
- 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>.
- ISO. *ISO/IEC 7498-1: Information technology - Open Systems Interconnection - Basic Reference Model: The Basic Model*. [S.l.], 1994.
- KALISKI, B. *TWIRL and RSA Key Size*. Maio 2003. <http://www.rsa.com/rsalabs/node.asp?id=2004>.
- LENSTRA, H. Factoring integers with elliptic curves. *Ann. Math.*, n. 126, p. 649–673, 1987.

MENEZES, A.; OORSCHOT, P. V.; VANSTONE, S. *Handbook of Applied Cryptography*. 1996. CRC Press.

MERKLE, R. Secure communications over insecure channels. *Communications of the ACM*, n. 21, p. 294–299, 1978.

NG, Y. *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>.

RIVEST, R.; SHAMIR, A.; ADLEMAN, L. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, n. 21, p. 120–126, 1978.

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

SAKAI, R.; OHGISHI, K.; KASAHARA, M. Cryptosystems based on pairing. In: *Symposium on Cryptography and Information Security – SCIS'2000*. Okinawa, Japan: [s.n.], 2000.

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.

SYLVERS, E. *Start-ups aiming for cheaper text messaging*. October 2007. Herald Tribune. <http://www.iht.com/articles/2007/10/07/business/phones08.php>.

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

APÊNDICE A - GLOSSÁRIO

- **Autenticidade** Garantia de que a entidade com que está-se comunicando é realmente a entidade desejada.
- **Cifrassinatura** Operação que combina assinatura e encriptação de uma mensagem em um único passo.
- **Confidencialidade** Garantia de que apenas os indivíduos permitidos tenham acesso à uma determinada informação.
- **Criptograma** Bloco de dados resultante da saída de algum processo criptográfico.
- **Emparelhamento** Função matemática bilinear que permite troca de posição das constantes que multiplicam os parâmetros sem que o resultado da função se altere.
- **Integridade** Garantia de que uma mensagem recebida não foi alterada no meio do caminho.
- **Irretratabilidade** Possibilidade de ciclaro convencer por exemplo um juiz de que fulano lhe assinou uma mensagem eletrônica.
- **Verificação** Operação que combina verificação e deciptação de uma mensagem em um único passo.

APÊNDICE B - DIAGRAMAS DE CLASSE DETALHADOS

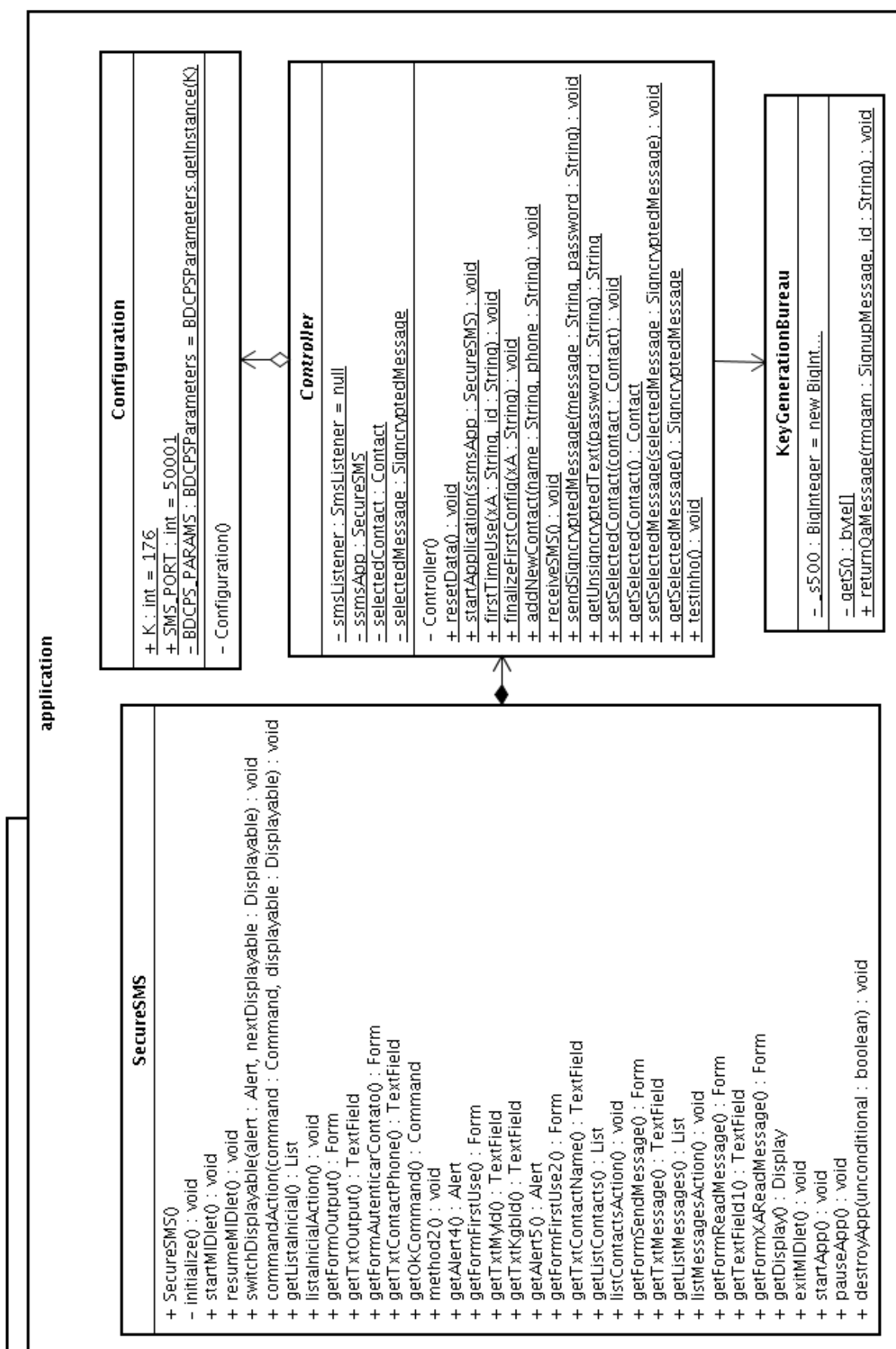


Figura 12: Diagrama de classes do pacote Application

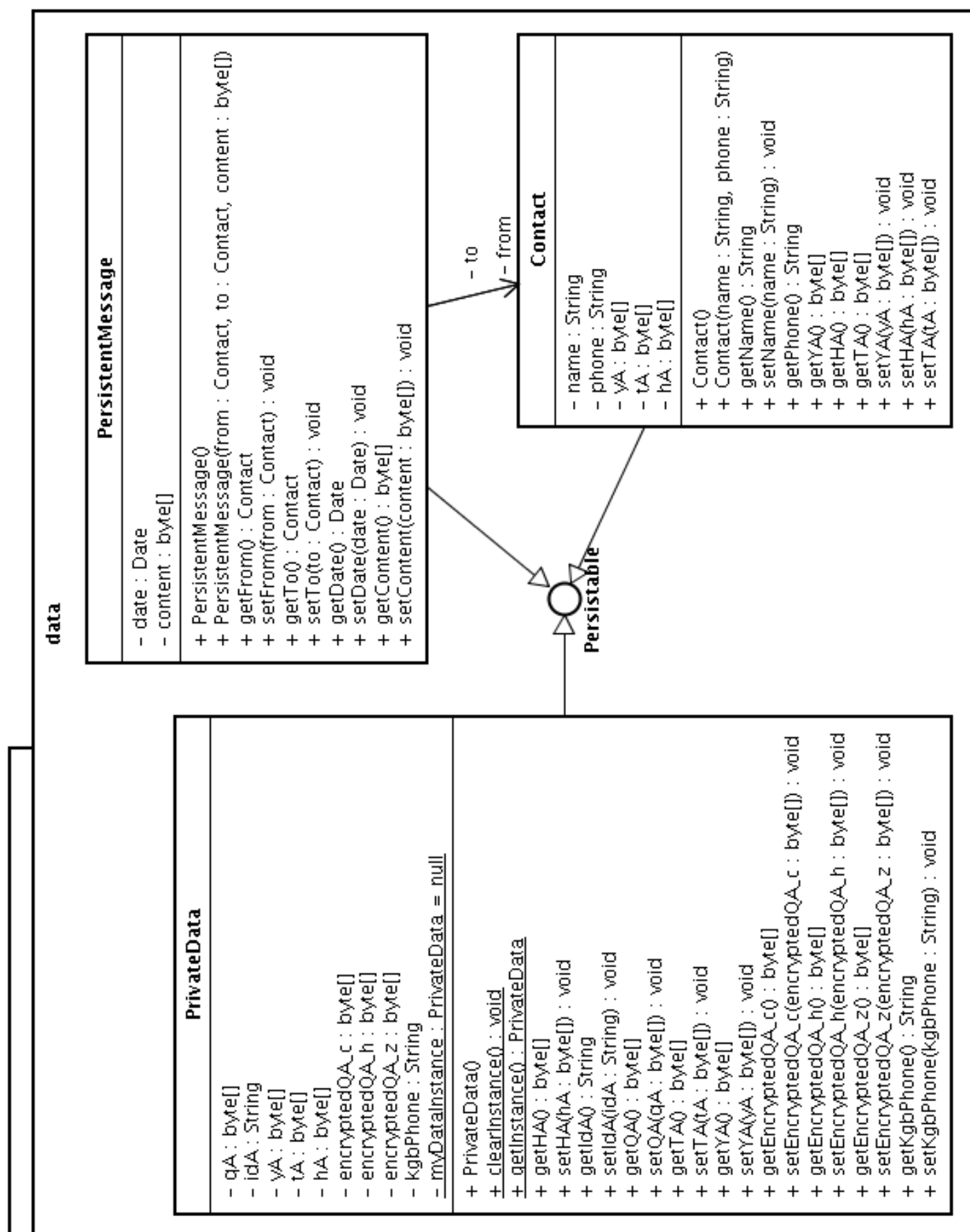


Figura 13: Diagrama de classes do pacote Data

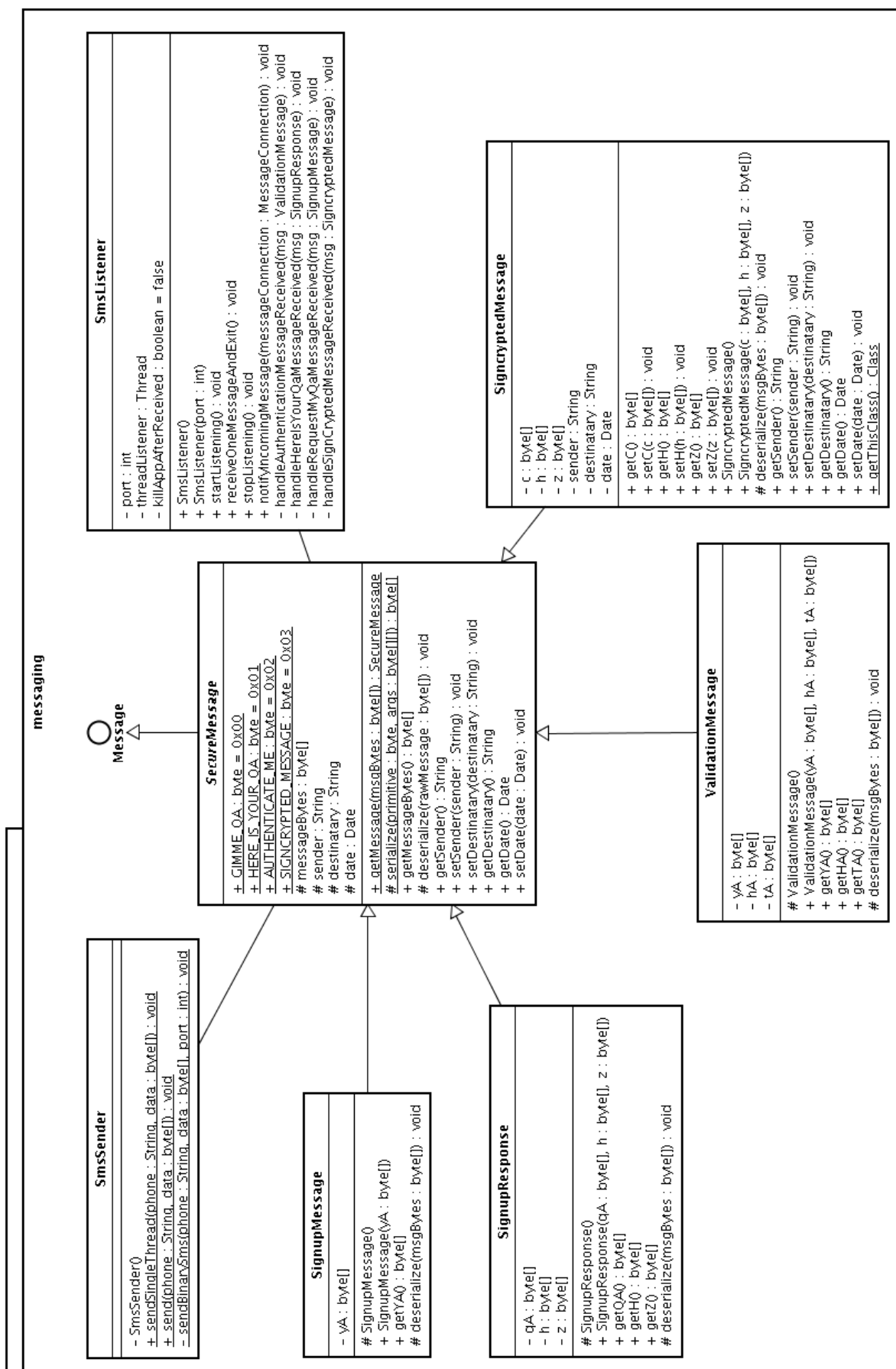


Figura 14: Diagrama de classes do pacote Messaging

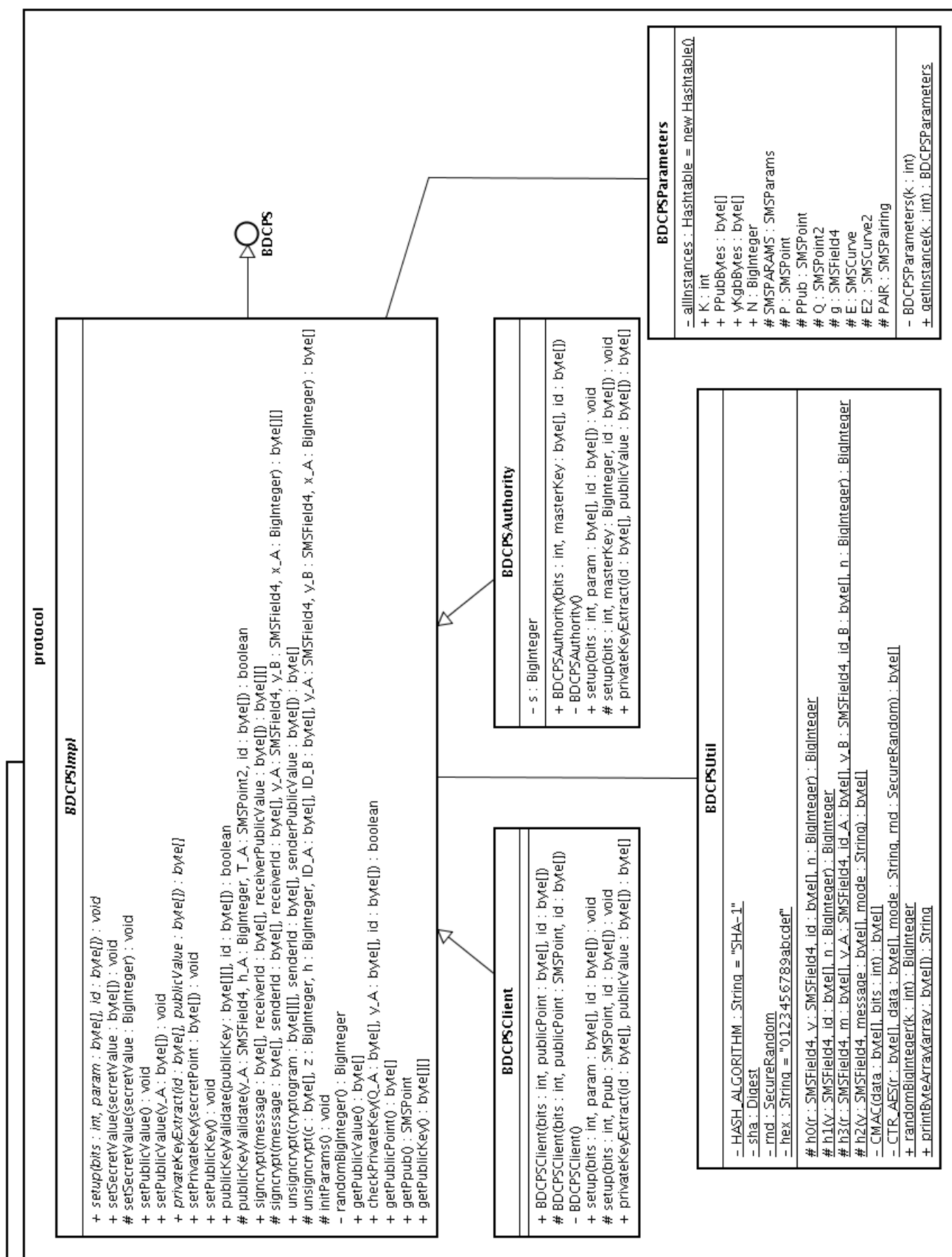


Figura 15: Diagrama de classes do pacote Protocol

APÊNDICE C - TELAS DO SISTEMA



Figura 16: Tela inicial do sistema

Primeiro uso

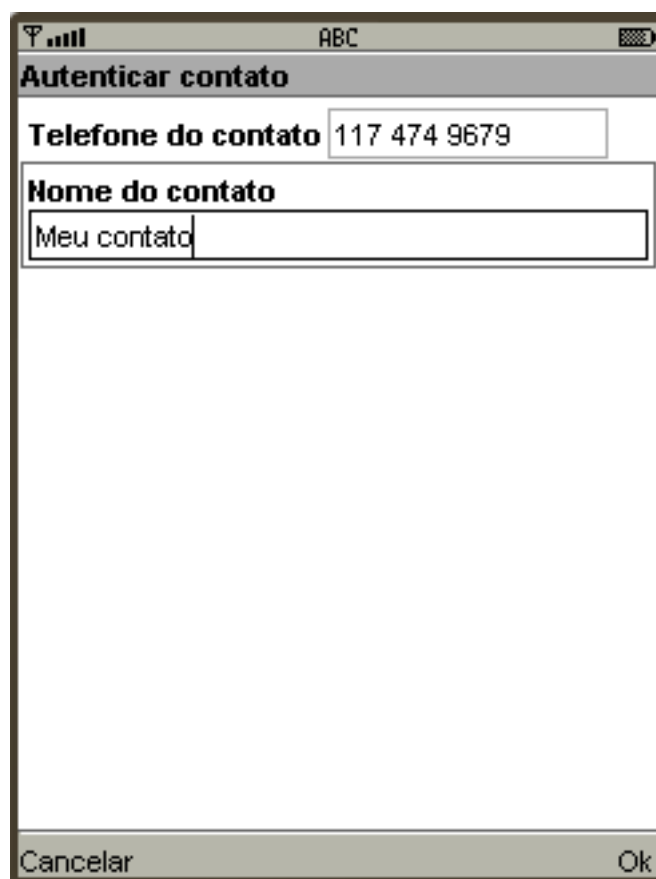
Senha secreta:

Entre com seu numero de telefone:
117 474 9679

Entre com o numero de telefone da KGB:
117 474 9679

Cancelar Ok

Figura 17: Tela de primeiro uso do sistema



The image shows a mobile application interface for authenticating a contact. At the top, there is a status bar with a signal strength indicator, the text "ABC", and a battery icon. Below this is a header bar with the title "Autenticar contato". The main content area contains two input fields. The first field is labeled "Telefone do contato" and contains the text "117 474 9679". The second field is labeled "Nome do contato" and contains the text "Meu contato". At the bottom of the screen, there is a footer bar with two buttons: "Cancelar" on the left and "Ok" on the right.

Autenticar contato

Telefone do contato 117 474 9679

Nome do contato

Meu contato

Cancelar Ok

Figura 18: Tela de validação de contatos



Figura 19: Tela de lista de contatos

The image shows a mobile application interface for sending a message. At the top, there is a status bar with a signal strength indicator, the text 'ABC', and a battery icon. Below this is a header bar with the title 'Enviar mensagem'. The main content area contains the following elements:

- A contact name: **Meu contato/1174749679**
- A label: **Mensagem:**
- A text input field containing the text: 'Meu teste!'
- A label: **Senha secreta:**
- A password input field containing masked characters: '*****'

At the bottom of the screen, there is a footer bar with two buttons: 'Cancelar' on the left and 'Ok' on the right.

Figura 20: Tela de envio de mensagem



Figura 21: Tela de lista de mensagens



Figura 22: Tela da caixa de entrada

APÊNDICE D - DESEMPENHO DA IMPLEMENTAÇÃO FINAL

Tabela 6: Medida de tempo de Signcrypt x tamanho da mensagem

Tam. Msg (bytes)	Emulador(ms)	Nokia E51(ms)	Nokia 6275(ms)
8	387,3	183,0	1440,8
28	393,6	144,8	1383,9
48	396,4	167,8	1369,6
68	388,1	148,8	1372,9
88	394,9	192,6	1394,0
108	397,2	157,9	1393,1
128	393,0	173,3	1381,0
148	397,3	149,4	1386,4
168	391,4	235,3	1391,3
188	394,7	147,4	1378,8
208	400,7	191,2	1395,9
228	393,9	151,6	1384,6
248	398,2	171,6	1387,1
268	396,6	147,7	1406,1
288	395,1	197,4	1405,6
308	398,4	159,9	1401,6
328	394,6	190,0	1410,8
348	395,5	189,7	1409,9

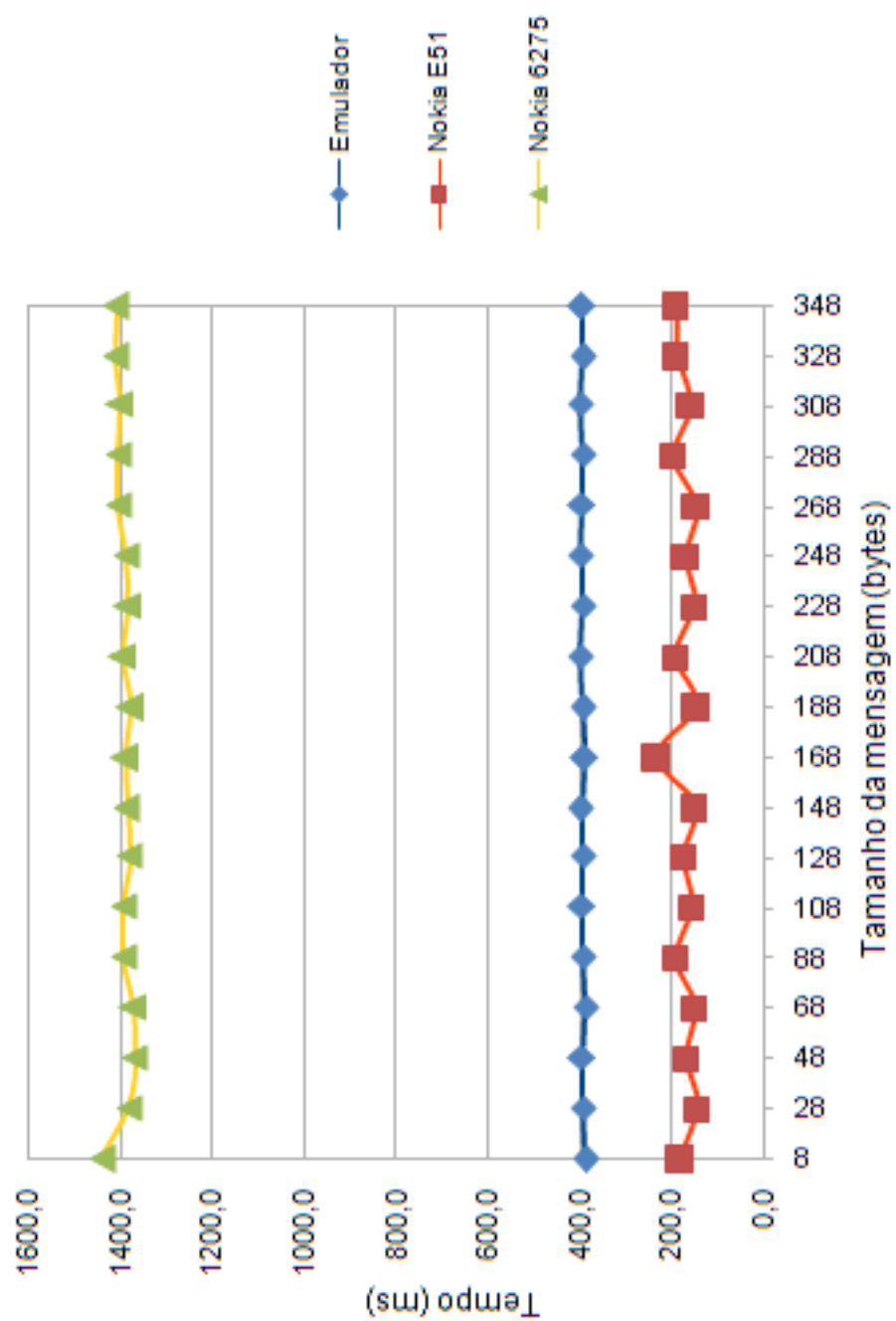


Figura 23: Gráfico: tempo de Signcrypt x tamanho da mensagem

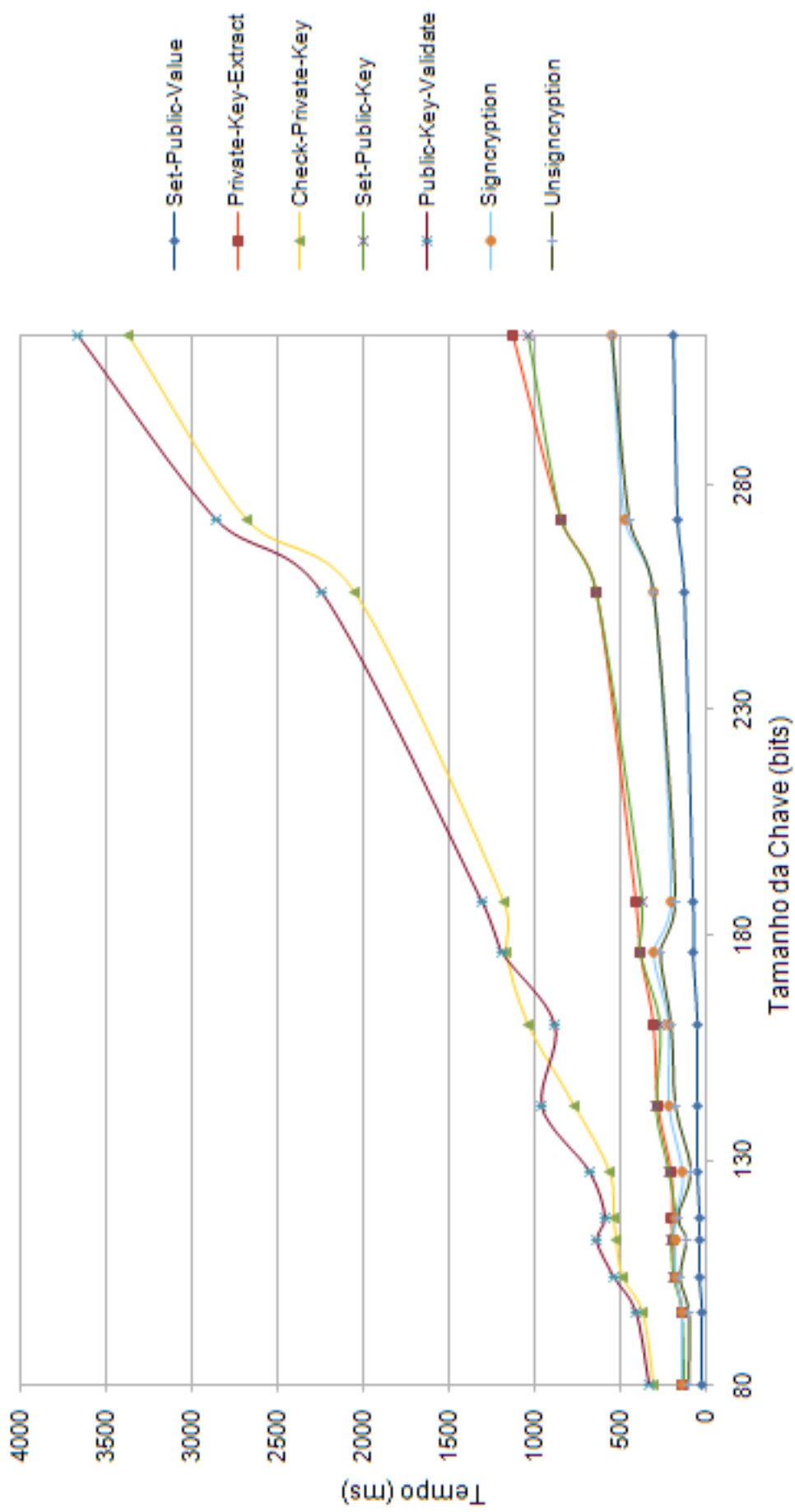


Figura 24: Gráfico: tempo das operações x tamanho da chave (Nokia E51)

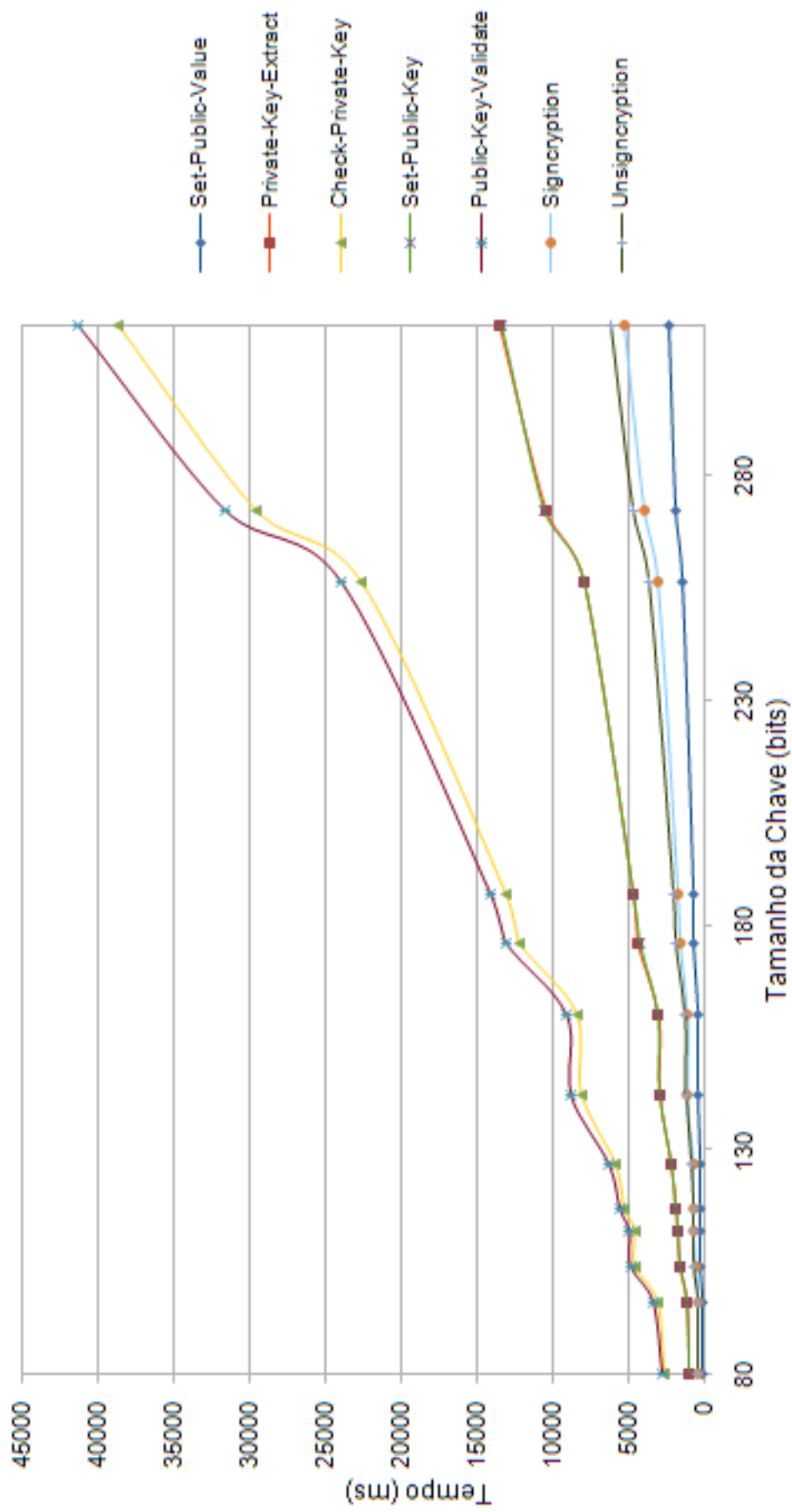


Figura 25: Gráfico: tempo das operações x tamanho da chave (Nokia 6275)

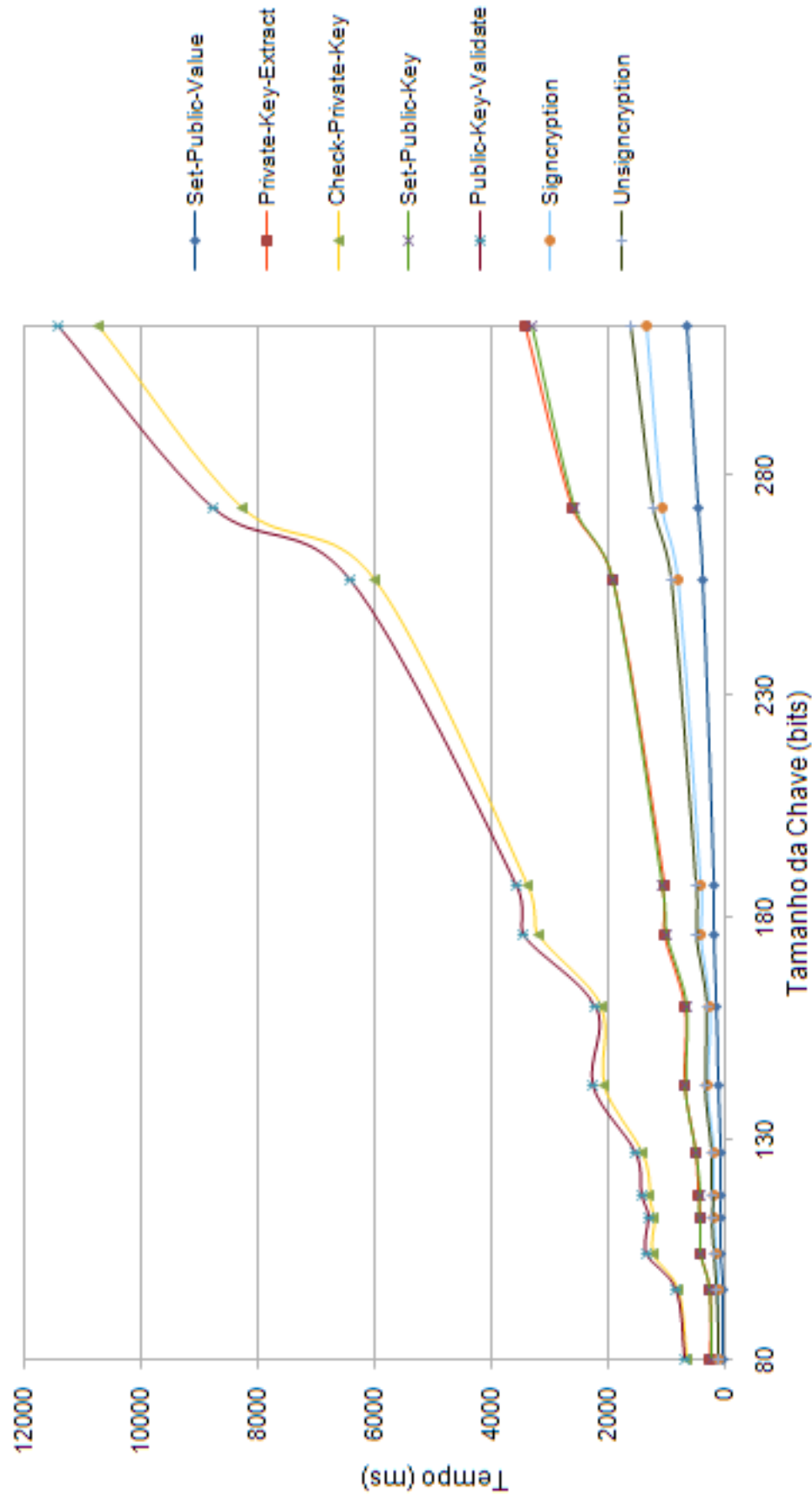


Figura 26: Gráfico: tempo das operações x tamanho da chave (Emulador WTK2.5.2)