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

APLICATIVO 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

APLICATIVO 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

Agradeço a... etc etc etc

RESUMO

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

ABSTRACT

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

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

SMS - *Short Message Service*

ECC - *Elliptic Curve Cryptography*

WTK - *Wireless Toolkit*

MIDP - *Mobile Information Device Profile*

CLDC - *Connected Limited Device Configuration*

KGB - *Key Generation Bureau*

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 XXX pessoas utilizam o SMS no mundo e XXX mensagens são trocadas anualmente.

A rede GSM (Global System for Mobile Comunnication), 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 email, 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 protegidas 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(**??**).

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

Atualmente não existem soluções universalmente adotadas para garantir segurança em mensagens SMS. Desse modo, 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 encontradas através dos mecanismos de busca na Internet eram escassas. No decorrer do ano, diversas alternativas foram surgindo. Listamos aqui algumas delas e resumimos suas principais características.

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.

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

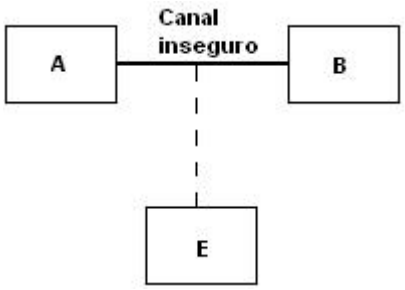

Figura 1: Canal inseguro

Ou, $\sim A$ poderia ser um web browser de um indivíduo A que está em processo de compra de um produto de uma loja online $\sim B$ 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 email 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 $\sim A$ 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 $\sim A$, ou poderia tentar personificar $\sim A$ para sacar fundos da conta de $\sim A$. 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.

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 (ORTIZ, 2002). Além disto, existem limitações de processamento no dispositivo celular, que podem

comprometer a usabilidade de um esquema criptográfico tradicional.

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

1.2 Objetivos

Considerando o cenário atual e suas limitações, 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.

Esse objetivo deve ser alcançado sem comprometer a usabilidade do serviço.

1.3 Metodologia

Nossa metodologia de pesquisa se divide basicamente em três vertentes. A primeira consistiu no estudo do cenário, o levantamento da necessidade de nossa solução e a especificação dos requisitos do sistema.

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 necessiadades 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 nossas necessiades.

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.

Organização O restante desta monografia organiza-se como se segue. No capítulo bla vla af

2 PRELIMINARES TEÓRICAS

2.1 Serviços de Segurança

Alguns dos principais serviços de segurança da informação estão listados abaixo:

- Confidencialidade: manter secretos os dados de todos a não ser aqueles autorizados a vê-los - mensagens enviadas por A para B não deveriam ser legíveis para .
- Integridade dos dados: assegurar que os dados não foram alterados por entidades não autorizadas significa - B deveria ser capaz de detectar quando dados enviados por A foram modificados por E.
- Autenticação da origem dos dados: confirmar a fonte dos dados - B deveria ser capaz de verificar que dados propositalmente enviados por A de fato foram originados por A.
- Autenticação da entidade: confirmar a identidade da entidade: B deveria se convencer da identidade da outra entidade em comunicação.
- Irretratabilidade: prevenir uma entidade de negar comprometimentos ou atos anteriores - Quando B recebe uma mensagem propositalmente 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. Conseqüentemente, eles podem usar um esquema de encriptação simétrica tal como o Data Encryption Standard (DES), RC4, ou o Advanced Encryption Standard (AES) para integrar confidencialidade.

Eles também podem usar um algoritmo de código de autenticação de mensagens (MAC) tal como o HMAC para reunir 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 - ENCk (m) para B. Ao receber c, B usaria a função de decriptação DEC e a mesma chave k para recuperar m = DECk (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 destes sistemas. Uma delas é o conhecido problema da distribuição de chave - a necessidade de um canal que seja ambos, 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 e conseqüentemente usa essas chaves para distribuir as informações de chaves para as entidades em comunicação quando requerido [1]. Soluções como esta podem ser bem apropriadas para ambientes onde uma autoridade central aceitável e confiável, mas é claramente impraticável 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 quando N se torna grande. Este problema pode ser aliviado usando serviços de uma terceira parte on-line que distribui as chaves conforme sãs 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 secreta. [2]

[1]Este 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.

[2]Esquemas de assinaturas digitais podem ser implementados usando técnicas de chave simétrica; contudo, estes esquemas geralmente impraticáveis conforme requerido seu uso de uma terceira parte confiável on-line ou novas informações de chaves para cada assinatura.

2.3 Criptografia de chave pública

2.3.1 Introdução

A noção de criptografia de chave pública, foi introduzida em 1975 por Diffie, Hellman e Merkle para resolver as deficiências da criptografia de chaves simétricas mencionadas anteriormente. Em contrate aos esquemas de chave simétrica, os esquemas de chave pública requerem apenas que as entidades em comunicação troquem informações de chaves que são autenticas (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 eB, e usa a função de encriptação ENC de uma esquema de chave pública para computar a cifra c = ENCeB (m). A então transmite c para B, que usa a função de decriptação DEC e sua chave privada dB para recuperar a mensagem clara: m = DECdB (c). A presunção é que um agressor com posse apenas de eB (mas não de dB) não consegue decriptar c. Observe que não há nenhuma necessidade de discrição de eB. É essencial apenas que A obtenha uma cópia autentica de eB - por outro lado A encriptaria m usando a chave pública eE 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 dA para computar a assinatura da mensagem: s = SIGNdA (m). Ao receber m e s, uma entidade B que tem uma cópia autentica da chave pública de A eA usa um algoritmo de assinatura de verificação para confirmar que s foi de fato gerado a partir de m e dA. Uma vez que dA é presumivelmente conhecido por A, B está assegurado de que a mensagem foi realmente originada por A. Ademais, uma vez que a verificação requer apenas quantidades não secretas m e eA, 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 $\sim m$ linha e afirmar que A assinou $\sim m$. Mesmo embora havendo nenhuma necessidade de segredo com relação à chave pública eA, é essencial que os verificadores devessem usar uma cópia autêntica de eA quando verificar assinaturas personificadamente geradas por A.

Deste modo, a criptografia de chave pública fornece soluções elegantes para os três problemas com criptografia de chave simétrica, chamados distribuição de chaves, administração de chaves e suporte à irretratabilidade. Deve-se notar que embora necessidade de um canal secreto para distribuição de chaves foi eliminado, implementar uma infra-estrutura de chave pública (PKI) para distribui e administrar chaves públicas pode ser um desafio formidável na prática. Também, operações em chave pública são normalmente significante-

mente mais lentas do que seus respectivos na criptografia de chave simétrica. Portanto, sistemas híbridos que beneficiam desde a eficiência dos algoritmos de chave simétrica e a funcionalidade dos algoritmos de chave pública são freqüentemente 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 é equivalente para resolver um problema computacional que é 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.

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. Em 1984, ElGamal descreveu a encriptação de chave pública baseada em DL e esquemas de assinatura. Desde então, muitas variantes destes esquemas foram propostas. Logo

em seguida apresentamos o esquema de encriptação básico de chave pública ElGamal e o Digital Signature Algorithm (DSA).

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 $gt \equiv 1(modp)$. 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 $y = g^x modp$. O problema de determinar x dados os parâmetros do domínio (p,q, g) e y é o problema do logaritmo discreto (DLP).

2.5 Curvas Elípticas

2.5.1 Histórico

O estudo das curvas elípticas, por matemáticos, data da metade do século XIX. Como conseqüência, encontramos hoje uma vasta literatura sobre o assunto. Em 1984, Hendrik Lenstra 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 dos números computacional.

A criptografia de chave pública foi concebida em 1976 por Whitfield Diffie e Martin Hellman. Sua primeira construção prática se seguiu em 1977 quando Ron Rivest, Adi Shamir e Len Adleman propuseram o protocolo agora tão conhecido RSA cuja segurança é baseada na intratabilidade do problema da fatoração de inteiros. A criptografia baseada em curvas elípticas (ECC) foi

descoberta em 1985 por Neal Koblitz e Victor Miller.

Os protocolos de ECC são mecanismos de criptografia de chave pública e fornecem as mesmas funcionalidades que o esquema proposto no RSA. Contudo, sua segurança é baseada na dificuldade de um diferente problema, o problema do logaritmo discreto em curvas elípticas (ECDLP), que é um pouco mais difícil de se resolver. Atualmente, os melhores algoritmos para resolver o ECDLP levam tempo completamente exponencial, enquanto que para o problema da fatoração de inteiros do RSA, o tempo é sub-exponencial. Isso significa que o nível de segurança desejado pode ser obtido com chaves significativamente menores que as usadas no RSA. Por exemplo, o nível de segurança determinado por uma chave de 160 bits em sistemas ECC é equivalente para o obtido usando-se RSA com chaves de 1024 bits. As vantagens obtidas com tamanhos de chaves menores são velocidade e uso eficiente de energia, largura de banda e armazenamento.

Desde então, grande quantidade de pesquisas começou a ser publicada na segurança e implementação eficiente em ECC. 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 $(G,*)$ consiste de um conjunto G com uma operação binária * : $GxG \rightarrow G$ satisfazendo as seguintes propriedades: (Associatividade) $a * (b * c)$ = $(a * b) * c$ para todos os $a, b, c \in G$ (Existência de uma identidade) Existe um elemento $e \in G$ tal que $a * e = e * a = a$ para todo $a \in G$. (Existência de inversos) Para cada $a \in G$, existe um elemento $b \in G$, chamado inverso de a, tal que $a * b = b * a = e$. (Comutatividade) $a * b = b * a$ para todos $a, b \in G$.

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 G é um conjunto finito, no caso em que o número de elementos em G é chamado a ordem de G. Por exemplo, seja p um numero primo, e Fp = 0,1,2,...,p-1 denota o conjunto dos inteiros módulo p. Então (Fp,+), 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 isso, $(F * p,\bullet)$, onde F*p denota os elementos diferentes de zero em Fp e a operação • 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 $(Fp, +, \bullet)$ é um campo finito, denotado mais sucintamente por Fp.

Agora, se G é um grupo finito multiplicativo de ordem n e $g \in G$, 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 $< g >= gi : 0 \leq i \leq t - 1$ de todas as

potências de g é ele próprio um grupo sobre a mesma operação como G, e é chamado um subgrupo cíclico de G gerado por G. Declarações análogas são verdadeiras se G é escrito aditivamente. Assim, a ordem de g G é o menor divisor positivo t de n tal que $tg = 0$, e $< g >= ig : 0 \leq i \leq t - 1$. 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 G. Por exemplo, com o os parâmetros do DL (p,q, g) definidos anteriormente, o grupo multiplicativo $(F * p, \bullet)$ é um grupo cíclico de ordem p-1. Ademais, $< g >$ é um subgrupo cíclico de ordem q.

2.5.3 Grupos em Curvas Elípticas

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

$$y^2 = x^3 + ax + b \tag{2.1}$$

onde a, b Fp satisfaz $4a^3 + 27b^2 = 0(modp)$. Um par (x, y), onde x, y Fp, é um ponto na curva se (x, y) satisfaz a equação 4.1. O ponto no infinito, denotado por ∞, também é considerado estar contido na curva. O conjunto de todos os pontos sobre E é denotado pro $E(Fp)$. Por exemplo, se E é uma curva elíptica sobre $F7$ com definida pela equação:

$$y^2 = x^3 + 2x + 4 \tag{2.2}$$

Então, os pontos sobre E são:

$$E(F7) = \infty, (0, 2), (0, 5), (1, 0), (2, 3), (2, 4), (3, 3), (3, 4), (6, 1), (6, 6).$$

Agora, há um método bem conhecido para somar 2 pontos numa curva elíptica $P : (x1, y1)$ e $Q : (x2, y2)$ para produzir um terceiro ponto na curvar

$R : (x3, y3)$. Veja a figura seguinte:

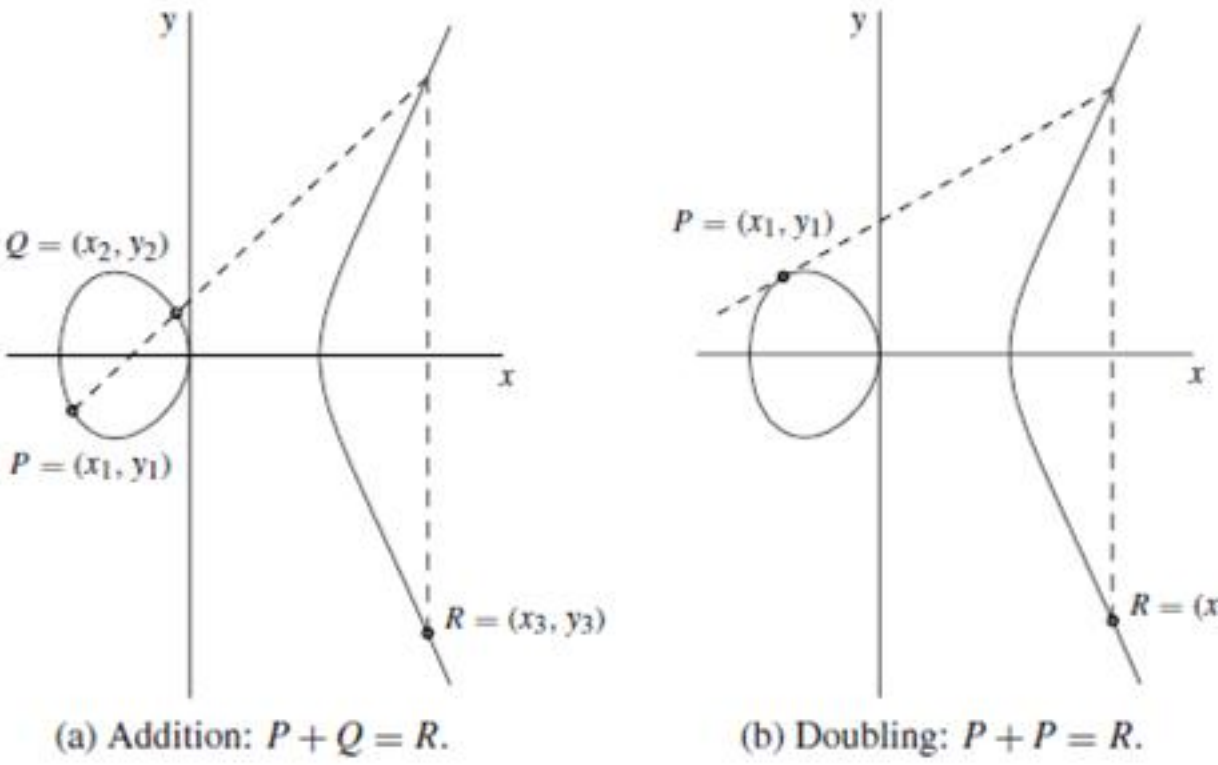

Figura 2: Soma em Curva Elíptica

A regra de adição requer algumas operações aritméticas (adição, subtração, multiplicação e inversão) in fp com as coordenadas $x1, y1, x2, y2$. Com esta regra de adição, o conjunto dos pontos E (Fp) 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.

2.5.4 Geração de chaves em curva elíptica

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

$< P >= \infty, P, 2P, 3P, ..., (n-1)P.$

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).

2.5.5 Encriptação em curva elíptica

Uma mensagem m é primeiramente representada como um ponto M, e então encriptada somando-se ela a kQ, onde k é um inteiro aleatoriamente selecionado, e Q é a chave pública do receptor. O emissor transmite os pontos $C1 = kP$ e $C2 = M + kQ$ para o receptor que usa sua chave privada d para calcular $dC1 = d(kP) = k(dP) = kQ$, e, em seguida, recupera $M = C2 - kQ$. Um agressor que deseja recuperar M precisa calcular kQ. Esta tarefa de computar kQ a partir dos parâmetros do domínio, Q, e $C1 = kP$, é um problema análogo ao problema de Diffie-Hellman em curvas elípticas.

2.6 Criptografia baseada em identidades e sistemas isentos de certificados

Shamir introduced the notion of identity-based (IB) cryptography (SHAMIR, 1984) in an attempt to mitigate the burden of a PKI. In an IB cryptosystem, private keys are not chosen by the users but rather issued by a trusted authority called the Key Generation Bureau (KGB) or Trust Authority (TA), and public keys are replaced by arbitrary strings representing users' identities, avoiding the need for certificates altogether. On the other hand, it has the drawback of

implicitly establishing a key escrow mechanism, since the KGB has the ability to recover confidential information from any user.

The concept of certificateless (CL) cryptosystems (**??**) was introduced to address the key escrow issue while avoiding the use of certificates and the need for a public-key infrastructure. The usual principle behind a CL scheme is to partition private keys into two components: an identity-based partial key (known to the KGB and thus otherwise subject to escrow) and one conventional albeit non-certified partial key (unknown to the KGB). This technique potentially combines the best features of identity-based and certificate-based cryptography, and indeed a number of certificateless encryption schemes derived from identity-based encryption algorithms have been successfully constructed and proven secure under certain assumptions. On the other hand, certificateless signatures remain much trickier to define and analyze, and as a consequence constructing certificateless signcryption schemes has been an elusive task only recently solved, though the only known such protocol can be hardly considered efficient. A signcryption scheme (ZHENG, 1997) is an integrated method to encrypt and sign a message in a more efficient way than to apply separately an encryption scheme and a signature scheme. The efficiency improvement may reside in the processing time, bandwidth occupation, key management, or any combination thereof; it may be simply a robust way to combine the two primitives so as to avoid deleterious interactions.

2.7 Emparelhamentos Bilineares

Identity-based cryptography became feasible when instantiated with the help of bilinear maps, or *pairings* for short (**??**BONEH; FRANKLIN, 2001), and has since attracted widespread attention due to its unusual properties. Pairings are formally defined as follows. Let k be a security parameter and n be a k-bit prime

number. Consider groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ of order n. We say that $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ are pairing groups if there exists a bilinear map (i.e. a pairing) $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ satisfying the following properties:

1. Bilinearity: $\forall (S, T) \in \mathbb{G}_1 \times \mathbb{G}_2$, $\forall a, b \in \mathbb{Z}_n$, $e(aS, bT) = e(S, T)^{ab}$.
2. Non-degeneracy: $\forall S \in \mathbb{G}_1$, $e(S, T) = 1$ for all $T \in \mathbb{G}_2$ iff $S = \mathcal{O}_{\mathbb{G}_1}$.
3. Computability: $\forall (S, T) \in \mathbb{G}_1 \times \mathbb{G}_2$, $e(S, T)$ is efficiently computable.

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, this algorithm chooses a k-bit prime number n, bilinear map groups $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ of order n supporting an efficiently computable, non-degenerate pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$, generators $P \in \mathbb{G}_1$, $Q \in \mathbb{G}_2$ and hash functions $h_0 : \mathbb{G}_T \times \{0,1\}^* \rightarrow \mathbb{Z}_n^*$, $h_1 : \{0,1\}^* \rightarrow \mathbb{Z}_n^*$. A master key $s\mathbb{Z}_n^*$ is also chosen, to which the public key $P_{pub} = sP \in \mathbb{G}_1$ is associated. The generator $g = e(P, Q) \in \mathbb{G}_T$ is also included among the public parameters which are $\texttt{params} = (k, n, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, P, Q, g, P_{pub}, e, h_0, h_1)$.
- **Private-Key-Extract:** takes as input entity A's identifier $\mathrm{ID}_A \in \{0,1\}^*$ and extracts A's identity-based private key $Q_A \leftarrow (h_1(\mathrm{ID}_A) + s)^{-1} Q \in \mathbb{G}_2$. Entity A can verify the consistency of this key by checking that $e(h_1(\mathrm{ID}_A)P + P_{pub}, Q_A) = g$. This setting is called the Sakai-Kasahara key style (SAKAI; KASAHARA, 2003).

- **Sign:** to sign $m \in \{0,1\}^*$ under the private key P_A, the signer picks $u\mathbb{Z}_n^*$ and computes

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

  The signed message is the triple $(m, h, S) \in \{0,1\}^* \times \mathbb{Z}_n^* \times \mathbb{G}_2$.

- **Verify:** given an identity ID_A, upon reception of (m, h, S) the verifier computes

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

  The verifier accepts the signed message iff $v = h$.

This scheme can be shown to be existentially unforgeable under adaptively chosen message attacks (EUF-IBS-CMA for short) in the random oracle model under the q-SDHP assumption (BARRETO et al., 2005, section 3.1). Notice that in this description we choose to define $P_{pub} \in \mathbb{G}_1$, $Q_A \in \mathbb{G}_2$ to avoid $\mathbb{G}_2$ arithmetic during verification, but an analogous description with $Q_{pub} \in \mathbb{G}_2$, $P_A \in \mathbb{G}_1$ and signed messages in $\{0,1\}^* \times \mathbb{Z}_n^* \times \mathbb{G}_1$ would be equally secure, while keeping the signature as short as possible in practice.

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 cifrassinatura de Zheng (ZHENG, 1997) em um esquema isento de certificados conforme proposto por (**??**). Este protocolo foi criado para atender às necessiades 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 $s \overset{R}{\leftarrow} \mathbb{Z}_n^*$ 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, $\texttt{params} = (k, n, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, P, Q, g, P_{pub}, h_0, h_1, h_2, h_3)$.

- **Set-Secret-Value:** dados `params`, o algoritmo toma $x_A \overset{R}{\leftarrow} \mathbb{Z}_n^*$ 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 $\mathrm{ID}_A \in \{0,1\}^*$, o identificador de A e o valor público $y_A \in \mathbb{G}_T$, e calcula a chave privada baseada em identidade de A, $Q_A \leftarrow (h_1(y_A, \mathrm{ID}_A) + s)^{-1} Q \in \mathbb{G}_2$. A entidade A consegue verificar a consistência desta chave checando se $e(h_1(y_A, \mathrm{ID}_A)P + P_{pub}, Q_A) = g$. 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 $u_A \overset{R}{\leftarrow} \mathbb{Z}_n^*$ e calcula

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

  A chave pública completa da entidade A é a tripla $(y_A, h_A, T_A) \in \mathbb{G}_T \times \mathbb{Z}_n^* \times \mathbb{G}_2$. Este 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. $r_A \leftarrow e(h_1(y_A, \mathrm{ID}_A)P + P_{pub}, T_A) y_A^{h_A}$

2. $v_A \leftarrow h_0(r_A, y_A, \mathrm{ID}_A)$

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 $u \overset{R}{\leftarrow} \mathbb{Z}_n^*$ e calcula

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

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

- **Unsigncrypt:** dada a chave pública do emissor $y_A \in \mathbb{G}_T$ previamente validade para a identidade 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. $v \leftarrow h_3(r, m, y_A, \mathrm{ID}_A, y_B, \mathrm{ID}_B)$

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 alguns serviços básicos de segurança como confidencialidade, integridade e autenticidade das mensagens SMS, permitindo ao usuário assinar, cifrar, decifrar e verificar mensagens enviadas por SMS. As soluções adotadas no projeto envolvem criptografia de chave pública, criptografia baseada em idententidade e também esquemas auto-certificados. Tal adoção deve deve dispensar a existência de um diretório de chaves públicas, uma vez que seria necessário o uso de certificados que ocupam muita banda do abiente restrito utilizado e encarece o processo.

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, sendo transparente à arquitetura interna da rede móvel. Essas duas qualidades tornam possível sua viabilidade em diversos cenários.

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

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

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

3.2 Métricas

A seguir, definimos as métricas desejadas.

- Nível de segurança: Desejamos que o nível de segurança de nosso protocolo seja equiparável ao nível de segurança do RSA usando chaves de 1024 bits, que pode ser calculado como 2 a 1024.
- Tempo de espera: Consiste nos tempos para cifrassinar e vericifrar uma mensagem. Baseando-se em aplicações já existentes e satisfazendo os requisitos de usabilidade de nosso projeto, estimamos que um inter-

valo de espera para processamento de uma mensagem de no máximo 5 segundos seja tolerável pelo usuário.

- Tamanho máximo de mensagens do protocolo: Consiste da soma dos bytes úteis da mensagem com os bytes de controle do algoritmo. Implementações SMS baseadas em *Sun Wireless Messaging API (WMA)* podem dividir uma única mensagem em, no máximo, 3 segmentos, totalizando 399 bytes binários (ORTIZ, 2002). Porém para evitar problemas devido à segmentação, decidimos que as mensagens do nosso protocolo deverão caber em apenas 1 segmento, o que nos dá um tamanho de 140 bytes binários para cada mensagem do protocolo.

- Tamanho das chaves privada/pública: Devido às limitações de banda, estabeleceu-se que cada o tamanho chave usada não deverá exceder 200 bits. No entanto, essa restrição não deve comprometer o nível de segurança desejado.

- Tamanho máximo de uma mensagem de texto a ser cifrassinada e enviada: Uma mensagem cifrassinada deverá caber em uma única mensagem do protocolo, cujo tamanho máixmo foi defindo em 140 bytes. Porém nem todos os bytes poderão ser usados para a mensagem, pois existirá um overhead do protocolo, devido à cabeçalhos e dados da assinatura. Estabelecemos então um tamanho máximo de 70 bytes para uma mensagem de texto. Desta forma, ficam reservados outros 70 bytes para o overhead e a assinatura.

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

[1]Mais tarde o leitor verá que optamos por um esquema que não usa certificados, ou seja,

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

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

Sendo assim, foi considerado o uso de criptografia em curvas elípticas com assinatura baseada em identidades, de acordo com o conceito proposto inicialmente por Shamir (SHAMIR, 1984). Aprofundando-se na especificação, percebeu-se ainda que a chave pública do usuário poderia ser estabelecida essencialmente a partir de sua identificação única no sistema, ou seja, seu próprio número de celular. Desse modo, a criptografia em curvas elípticas baseada em identidades com emparelhamentos bilineares parecia ser capaz de atender aos requisitos do nosso aplicativo.

---

esta métrica será atendida com 0 bits de tamanho de certificado

4 ANÁLISE DE REQUISITOS DO SISTEMA

4.1 Casos de uso

4.1.1 Diagrama

4.1.2 Atores

1. Ator 1:

4.1.3 Fluxos

1. Cadastrar-se no sistema

   (a) Descrição: Novo usuário deseja usar o sistema pela primeira vez e precisa efetuar as configurações necessárias para poder receber sua chave privada.

   (b) Evento Iniciador: Usuário seleciona a opção de primeiro uso do sistema

   (c) Atores: Usuário, KGB.

   (d) Pré-condição: Sistema de SMS seguro apresentando sua tela inicial.

   (e) Seqüência de Eventos:

      i. Usuário seleciona o botão de primeiro uso.

      ii. Sistema pede a entrada de uma nova senha privada do usuário.

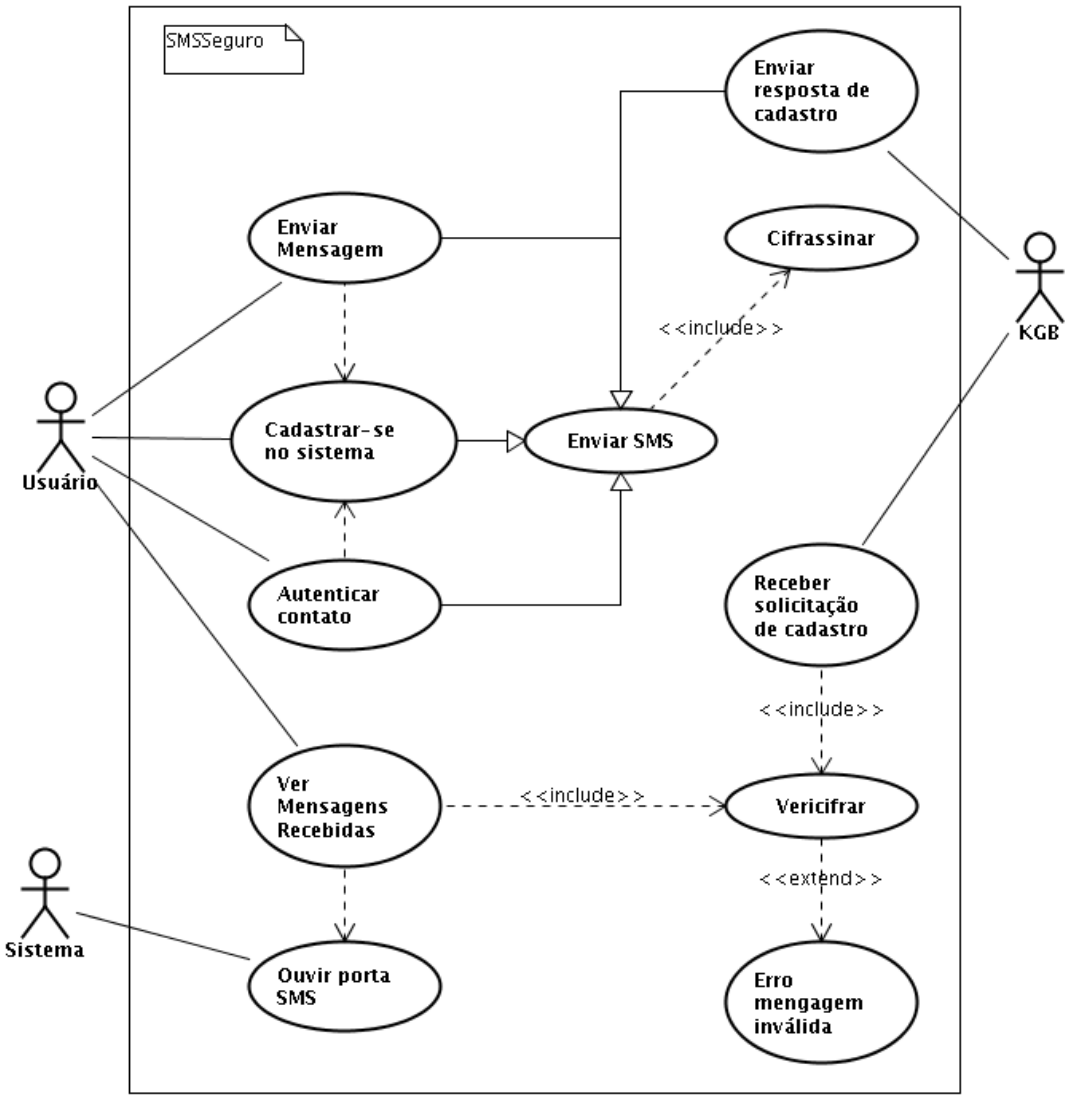

Figura 3: Diagrama de casos de uso do sistema

iii. Usuário cadastra uma nova senha no sistema e confirma.

iv. Sistema exibe notificação de envio de mensagem de controle para a KGB.

v. Usuário confirma o envio e sistema transmite a mensagem.

vi. Ao receber a mensagem, a KGB gera a chave privada do usuário e retorna uma mensagem segura contendo a chave gerada.

vii. O sistema do usuário recebe a mensagem da KGB contendo sua chave privada.

viii. Sistema pede novamente a senha do usuário e extrai a chave privada do usuário

ix. O sistema verifica se a chave privada recebida é válida.

x. O sistema armazena a nova chave no celular.

(f) Pós-Condição: Sistema volta para a tela inicial e usuário está apto a autenticar novos contatos para trocar mensagens.

(g) Extensões:

i. 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.

2. Autenticar novo contato

(a) Descrição: Quando um usuário quiser trocar mensagem com um contato que ainda não foi autenticado pelo Sistema de SMS Seguro deverá requisitar sua autenticação.

(b) Evento Iniciador: Usuário deseja autenticar um novo contato para enviá-lo uma mensagem.

(c) Atores: Usuário que quer se comunicar, usuário recebedor da mensagem.

(d) Pré-condição: Sistema exibe tela inicial.

(e) Seqüência de Eventos:

i. Usuário seleciona a opção de autenticar novo contato.

ii. Usuário insere o número do telefone do novo contato e seleciona ok.

iii. Sistema exibe notificação de envio de SMS para o contato informado.

iv. Usuário confirma o envio da mensagem SMS.

v. Sistema envia requisição de autenticação para o novo contato.

(f) Pós-Condição: Sistema volta para a tela inicial.

(g) Extensões:

i. Sistema exibe notificação de erro no envio da mensagem caso o serviço de envio esteja indisponível. (Passo v).

3. Enviar Mensagem

(a) Descrição: Usuário deseja compor e enviar uma nova mensagem SMS para outro usuário.

(b) Evento Iniciador: Usuário seleciona botão de envio de nova mensagem.

(c) Atores: Usuário emissor da mensagem.

(d) Pré-condição: Sistema de SMS seguro apresentando sua tela inicial.

(e) Seqüência de Eventos:

i. Usuário emissor da mensagem seleciona botão de envio de mensagem.

ii. Sistema exige que o usuário indique o destinatário da mensagem.

iii. Usuário seleciona o destino da lista de contatos exibida.

iv. Usuário compõe a mensagem a ser enviada

v. Usuário confirma o envio da mensagem para o destinatário escolhido.

vi. Sistema exibe notificação de envio da mensagem.

(f) Pós-Condição: A notificação de mensagem enviada é exibida e o sistema retorna à tela inicial.

(g) Extensões:

i. Sistema exibe notificação de erro no envio da mensagem caso o serviço de envio esteja indisponível. (Passo v).

(h) Inclusões

i. Sistema busca todos os contatos da lista de contatos do celular para exibi-los.

ii. Caso de uso 4.

4. Recepção de Mensagem

(a) Descrição: Quando uma nova mensagem chega no celular o sistema deve captá-la e fazer seu tratamento.

(b) Evento Iniciador: Chega uma nova mensagem do Sistema de SMS Seguro no celular de um usuário.

(c) Atores: Sistema operacional.

(d) Pré-condição: Celular do usuário ligado.

(e) Seqüência de Eventos:

i. Celular recebe a nova mensagem e a coloca na fila.

ii. Sistema operacional do celular capta a mensagem e requisita ao usuário que inicialize a aplicação caso ela não esteja em execução.

iii. O sistema identifica a primitiva da mensagem e trata de forma correspondente.

(f) Pós-Condição: a mensagem está processada e o sistema está exibindo a tela inicial.

(g) Extensões:

i. Sistema trata mensagem cifrada e assinada. (Passo iii)

ii. Sistema trata mensagem de autenticação de usuário.(Passo iii)

iii. Sistema trata mensagem de pedido da chave privada. (Passo iii)

iv. Sistema trata mensagem de entrega de chave privada. (Passo iii)

5. Encriptação/Assinatura de Mensagem

(a) Descrição: Usuário escreveu uma mensagem para alguém e deseja cifrá-la e assiná-la.

(b) Evento Iniciador: Usuário requisita envio de mensagem cifrada e assinada ao sistema.

(c) Atores: Usuário que deseja enviar uma mensagem segura.

(d) Pré-condição: Usuário está com a mensagem pronta na tela de envio.

(e) Seqüência de Eventos:

i. Usuário seleciona a opção de enviar a mensagem.

ii. Sistema cifra e assina a mensagem e exibe tela de confirmação de envio.

iii. Usuário confirma o envio e sistema transmite a mensagem segura.

(f) Pós-Condição: Sistema volta para a tela inicial.

6. Verificar Mensagem

(a) Descrição: Usuário deseja visualizar uma mensagem na sua caixa de entrada.

(b) Evento Iniciador: Usuário abre a caixa de entrada do sistema.

(c) Atores: Usuário.

(d) Pré-condição: Sistema de SMS seguro apresentando mensagens recebidas na caixa de entrada. Seqüência de Eventos:

i. Usuário escolhe a mensagem que deseja visualizar e seleciona ok.

ii. Sistema verifica e decifra a mensagem.

iii. Sistema exibe a mensagem clara para que o usuário possa lê-la.

(e) Pós-Condição: Sistema exibindo mensagem clara para o usuário.

4.2 Requisitos funcionais

4.2.1 Requisito A

Introdução/Propósito: TODO TODO

Estímulo/Resposta: TODO TODO TODO

Detalhamento: TODO TODO TODO

4.2.2 Requisito B

Introdução/Propósito: TODO TODO

Estímulo/Resposta: TODO TODO TODO

Detalhamento: TODO TODO TODO

4.2.3 Requisito C

Introdução/Propósito: TODO TODO

Estímulo/Resposta: TODO TODO TODO

Detalhamento: TODO TODO TODO

4.3 Requisitos não-funcionais

4.3.1 Usabilidade

O tamanho da chave privada não deve prejudicar a usabilidade do software. O usuário deve ser capaz de digitá-la em menos de 60 segundos. O método de entrada das mensagens deve ser semelhante ao dos aparelhos celulares convencionais.

4.3.2 Desempenho

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

4.3.3 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).

4.3.4 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.

4.3.5 Compatibilidade

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

4.3.6 Portabilidade

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

4.3.7 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.

5 ESCOLHA DE UM ESQUEMA CRIPTOGRÁFICO ADEQUADO

5.1 Por quê usar 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 do RSA é:

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).

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 algoritmo

de criptografia baseada em identidades.

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

5.2.1 Testes de viabilidade

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

O desempenho observado inicialmente foi insatisfatório, não atendendo aos requisitos de usabilidade estabelecidos na especificação. Foram feitas tentativas de melhoria do desempenho, como variação do tamanho das chaves, uso de diferentes funções de emparelhamento (Ate, Eta)(FREEMAN; SCOTT; TESKE, 2006), e implementações com diferentes bibliotecas que fornecessem a classe *BigInteger* - a implementação da Sun se mostrou mais eficiente do que a implementação da Bouncy Castle. Algumas adaptações no esquema em si foram feitas, como inversão da ordem das curvas utilizadas.

Os melhores resultados obtidos são apresentados na tabela 1.

Tabela 1: Testes com BLMQ

Operação	Tempo (s)
Inicialização das classes	128.9
Emparelhamento Eta	4.2
Emparelhamento Ate	3.9

A implementação inicial tomava muito tempo computacional para inciali-

zar as classes usadas pelo esquema, pois muitos cálculos eram feitos. Além disso, observou-se que as operações de emparelhamento são muito custosas. Emparelhamentos são extensamente utilizados pelo BLMQ. A operação de signcrypt consome....

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

5.3 BDCPS

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

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

O esquema proposto integra esquemas preexistentes como as assinaturas BLMQ e Schnorr (SCHNORR, 1991) e o esquema isento de certificados de Zheng(ZHENG, 1997). Neste esquema, a geração das chaves dos usuários dispensa a necessidade de uma autoridade certificadora e a utilização de

certificados para validar sua chave pública. Estas características implicaram em importantes, mas não únicas, melhorias em relação ao esquema anteriormente implementado e serão discutidas na seção "Análise da Proposta". Por criptografia convencional entende-se o fato de o usuário poder escolher seu par de chaves não certificado, ou seja, ele escolhe apropriadamente uma chave privada e gera sua chave pública a partir dela. Desse modo, somente o usuário gerador de seu par de chaves convencional conhece sua chave privada, e este fato elimina a possibilidade de "Key Escrow", que é o comprometimento da chave e conseqüentemente das mensagens com ela encriptadas. O par de chaves não certificado é combinado com a solução de criptografia baseada em identidades para que seja posteriormente validado por outro usuário do sistema.

5.3.1 Vantagens do esquema

Dentre as operações realizadas nos diversos algoritmos, a que apresenta maior custo computacional é a operação de emparelhamento bilinear. 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.

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 veri-

cifração estão de acordo com as métricas estabelecidas e bem mais eficientes em relação ao esquema inicialmente adotado.

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

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

5.4 Análise dos

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

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

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

6 ESPECIFICAÇÃO E PROJETO

6.1 Arquitetura

6.2 Classes

6.2.1 Diagrama

6.2.2 Descrição

6.3 Especificação do protocolo

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

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 0x50.
- Byte 2: Byte fixo, identifica a primitiva SignupMessage. Valor 0x00.

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

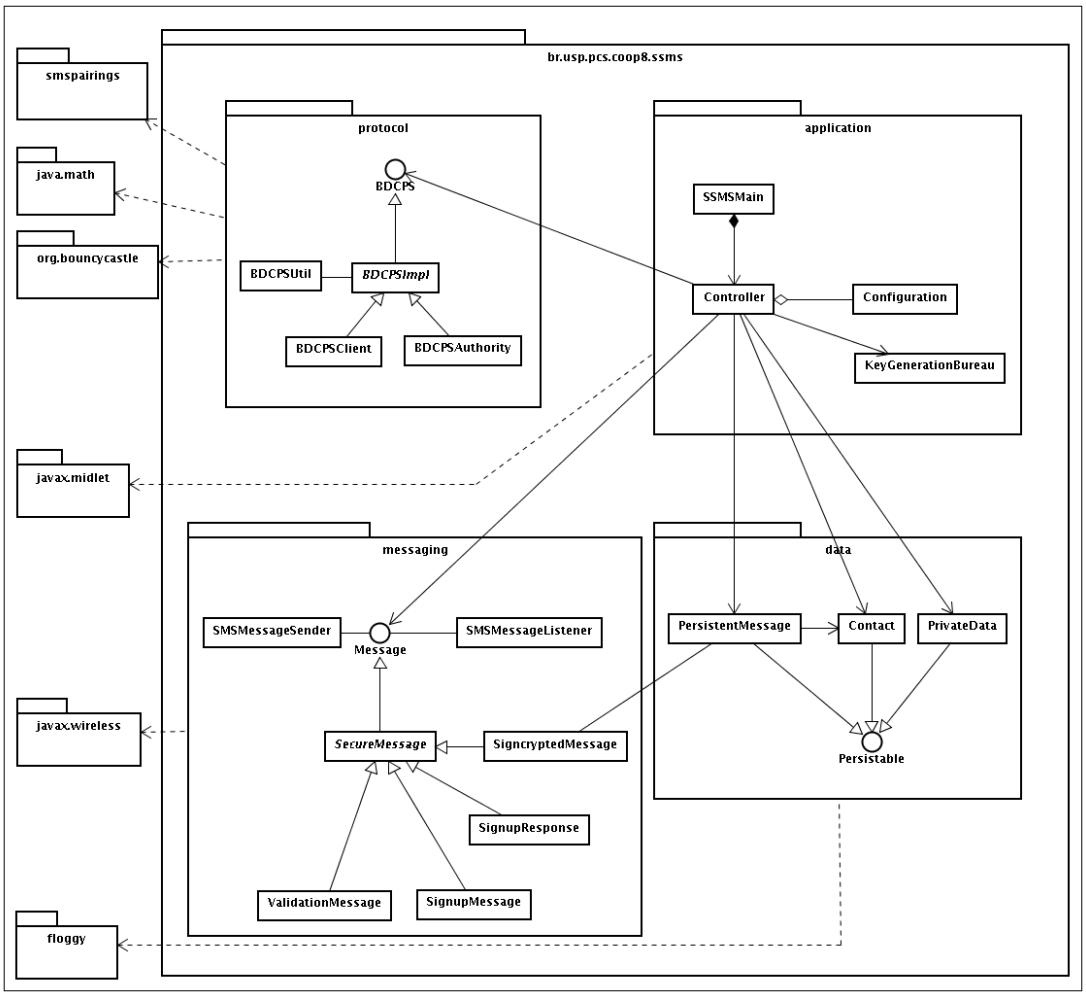

Figura 4: Diagrama de classes do sistema

- 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.

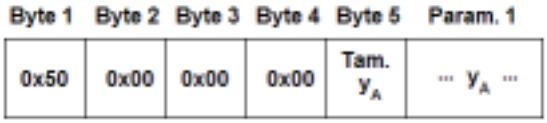

Figura 5: SignupMessage

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

- Byte 1: Byte fixo que identifica uma mensagem do nosso protocolo. Valor 0x50.
- Byte 2: Byte fixo, identifica a primitiva 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	0x00	0x00	Tam. c	Tam. h	Tam. z	... c ...	... h ...	... z ...

Figura 6: SignupResponse

6.3.3 ValidationMessage

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

- Byte 1: Byte fixo que identifica uma mensagem do nosso protocolo. Valor 0x50.
- Byte 2: Byte fixo, identifica a primitiva 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.

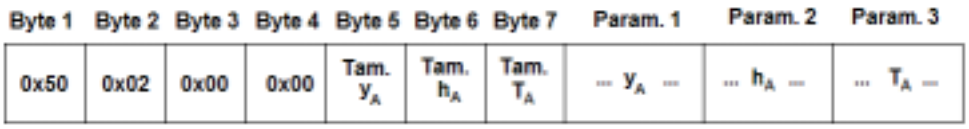

Figura 7: ValidationMessage

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 0x50.
- 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.

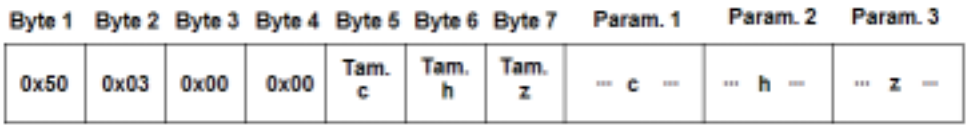

Figura 8: SignCryptedMessage

6.4 Escolha de parâmetros

6.4.1 Escolha de curvas elípticas adequadas

6.4.2 Escolha do tamanho de chave padrão

Como inicialmente desejávamos um nível de segurança equiparável ao do RSA, foi escolhido o número 176 como tamanho de chave, pois este fornece um nível de segurança de $176/2 = 88$[2], ou seja, um pouco superior ao nível de segurança do RSA-1024, que é de XXX.

[2]Apesar de termos escolhido este valor padrão, a aplicação pode ser facilmente alterada para trabalhar com os seguintes valores:

6.4.3 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 recebida[3].

Sendo assim, escolhemos arbitrariamente o valor 50001 para usar como a porta de nossa aplicação. A aplicação envia e se registra para escutar mensagens nesta porta.

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

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 para controle de versão e coordenação do trabalho em equipe, hospedado no servidor do GoogleCode.

7.2 Bibliotecas usadas

7.2.1 SMSPairings

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

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 framwork brasileiro para persistência de dados em ambiente J2ME. Foi essencial para nosso projeto para persistir objetos como mensagens crifrassinadas recebidas, contatos validados e dados do protocolo, como a chave privada parcial Q_A;

7.3 Problemas encontrados

7.4 Telas do sistema

Abaixo apresentamos algumas telas do sistema, depois de implementado.

Figura 9: Tela inicial do sistema

Figura 10: Tela de primeiro uso do sistema

Figura 11: Tela de adicionar contato

Figura 12: Tela de enviar mensagem

Figura 13: Tela de ler mensagem

8 RESULTADOS

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

8.0.1 Problemas com operadoras distintas

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 A acrescentamos gráficos demonstrando como estes tempos variam de acordo com o tamanho da chave usado, além de um gráfico demonstrando como o tempo da operação de cifrassinar uma mensagem varia conforme o tamanho da mensagem.

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

Operação	Tempo Nokia E51 (s)	Tempo Nokia 6275 (s)	Emulador WTK 2.5.2
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

8.2 Comparação com outras soluções

9 CONCLUSÃO

Ao longo do trabalho, foi possível observar que criptografia em curvas elípticas é a melhor alternativa para criptografia em ambientes com restrições, como dispositvos móveis.

9.1 Análise dos resultados

O sistema de modo geral funcionou de acordo com o esperado, satisfazendo nossos requisitos e métricas. Os resultados apresentados mostram tempos excelentes, atingindo assim as métricas de usabilidade desejadas, não existe a sensação de lentidão durante o uso do sistema.

Além disto, o nível de segurança desejado também foi alcançado. Com uma chave de 176 bits, temos um nível de segurança equivalente ao do RSA com 1024 bits.

9.2 Sumissão de artigo ao WTICG'08

9.3 Possíveis desenvolvimentos futuros

REFERÊNCIAS

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

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

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

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

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

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

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

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

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

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

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

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

ZHENG, Y. Digital signcryption or how to achieve 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 – DESEMPENHO DA IMPLEMENTAÇÃO FINAL

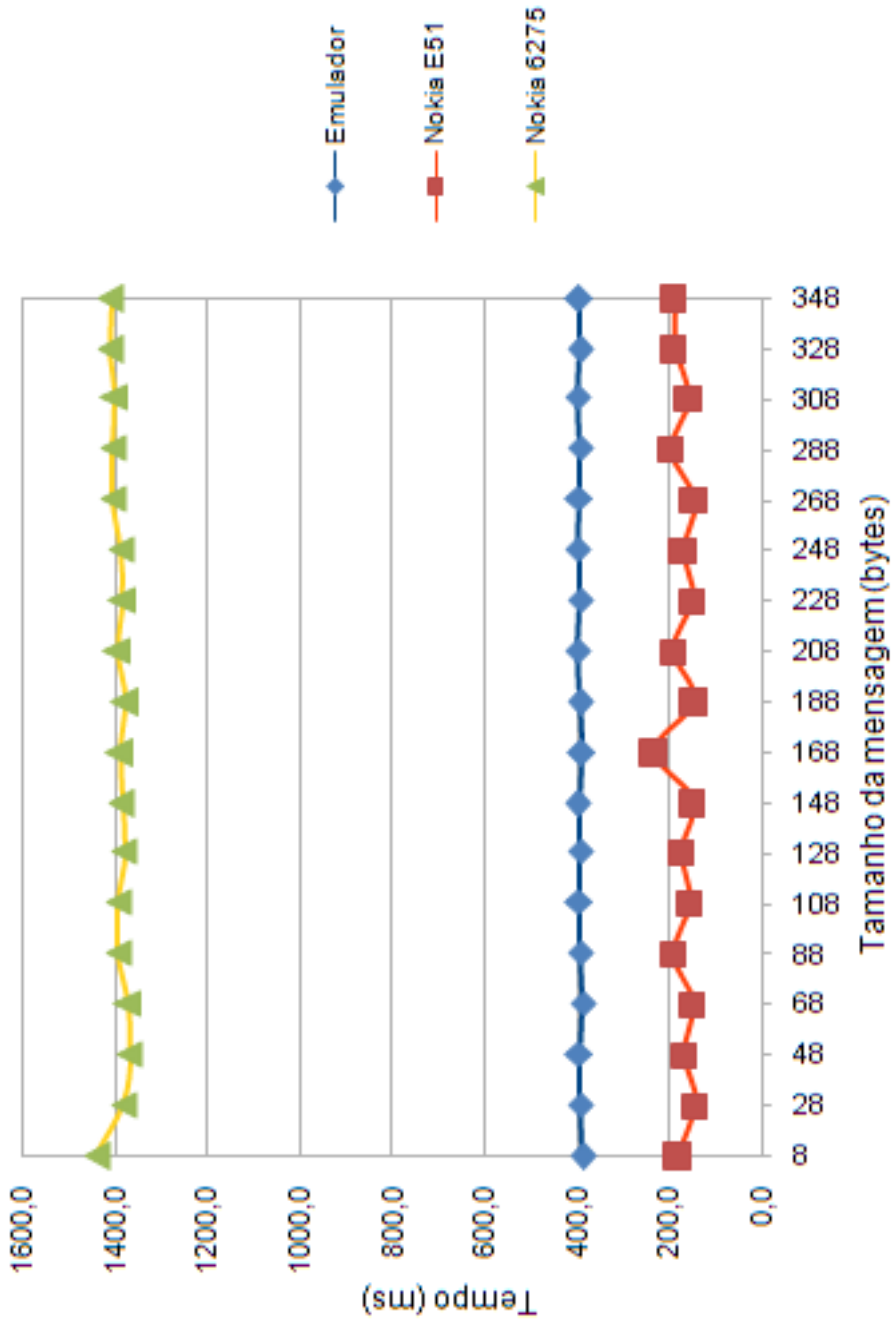

Figura 14: Variação do tempo de Signcrypt enquanto varia-se o tamanho da mensagem

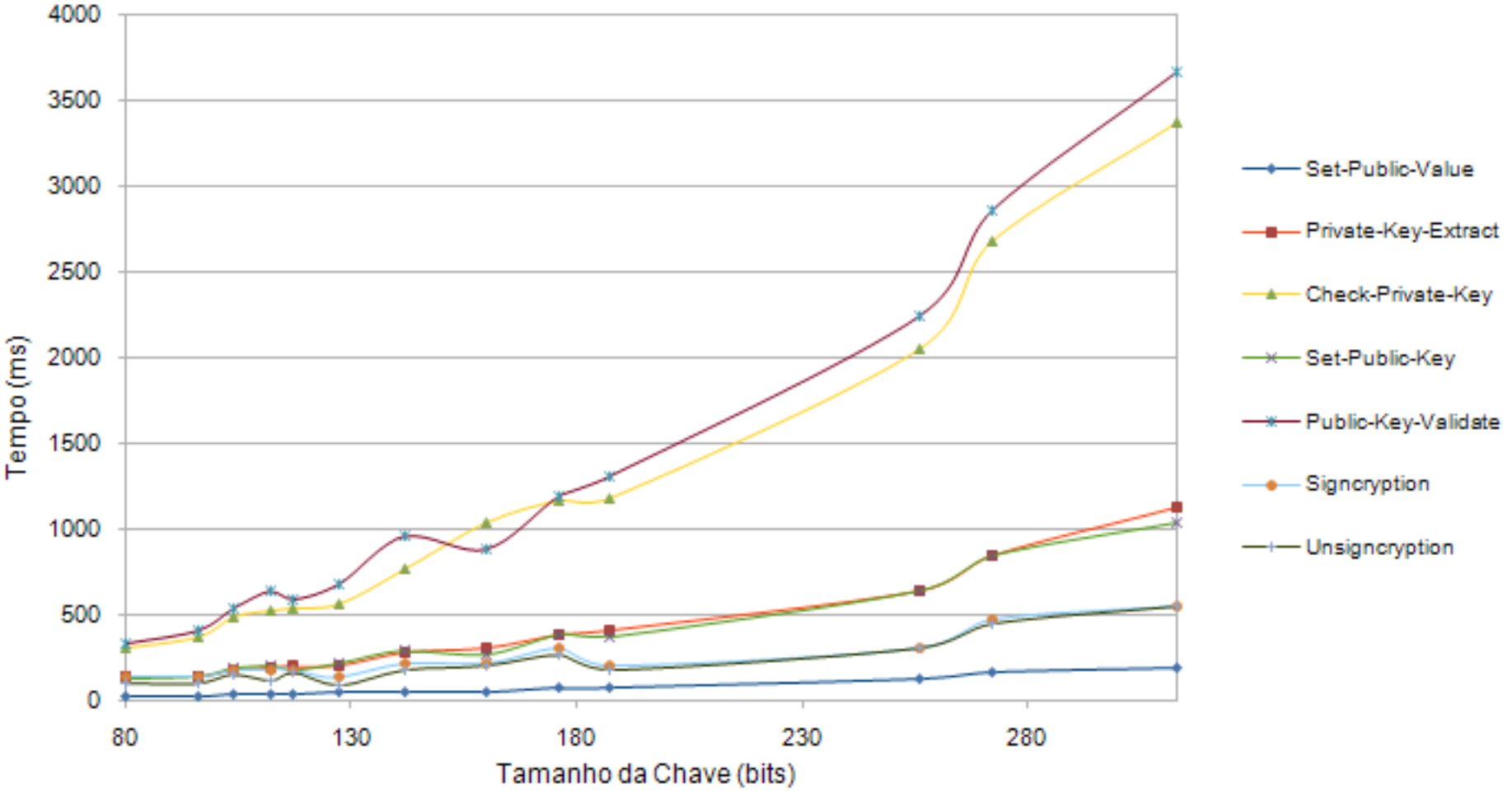

Figura 15: Variação do tempo das operações enquanto varia-se o tamanho da chave no celular Nokia E51

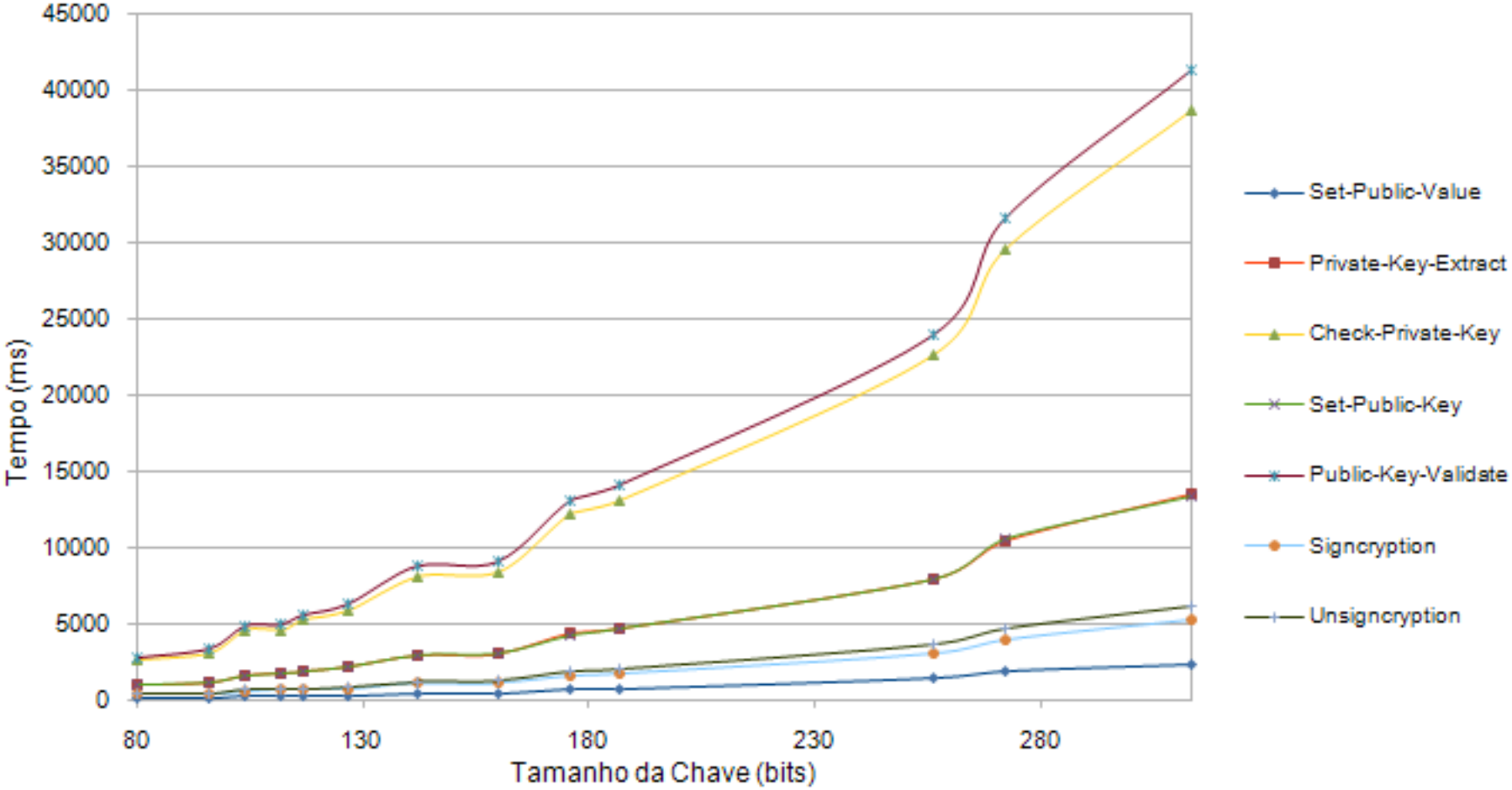

Figura 16: Variação do tempo das operações enquanto varia-se o tamanho da chave no celular Nokia 6275

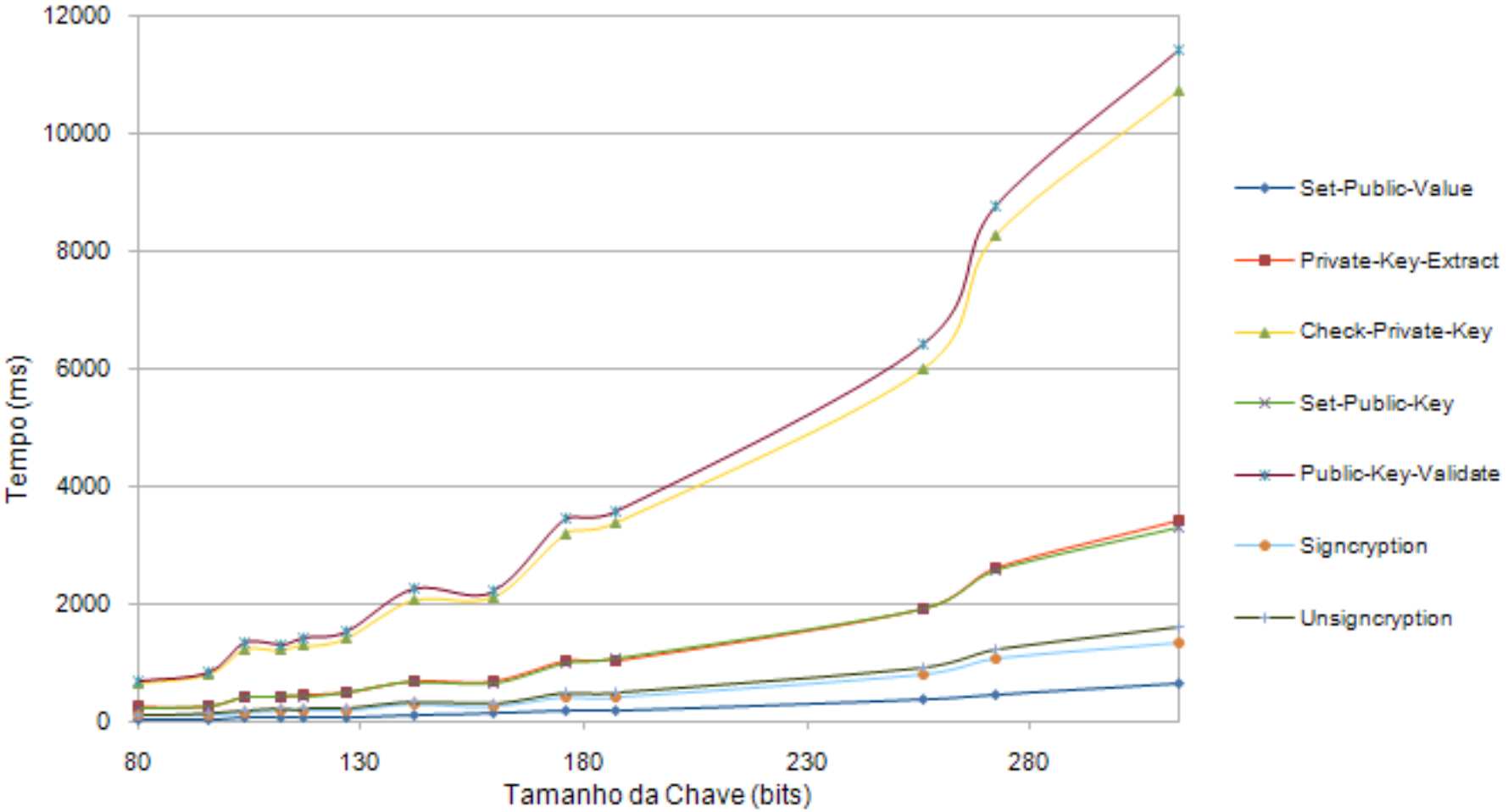

Figura 17: Variação do tempo das operações enquanto varia-se o tamanho da chave no emulador do WTK 2.5.2