

Protocolo MobiS: Uma solução para o Desenvolvimento de Aplicações Seguras para Dispositivos Móveis

Bringel Filho¹, Katy Magalhães², Windson Viana¹, Javam Machado¹ e Rossana Andrade¹

¹Mestrado em Ciência da Computação (MCC) – Universidade Federal do Ceará (UFC)
Fortaleza – CE – Brasil

²Programa de Pós-Graduação em Engenharia Elétrica (PPGEE)
Universidade Federal do Ceará (UFC)

{bringel, katy, windson, javam, rossana}@lia.ufc.br

Abstract. *With the evolution of the mobile computing, portable devices have emerged in the market with the possibility of connection to IP networks using wireless technologies. The migration of fixed terminals to wireless mobile devices and the increasing availability of information for such devices impose new challenges in the development of applications. The security and the portability are examples of these challenges that constitute the motivation for this work. The main goal is to introduce a protocol, called MobiS, to the development of secure applications for data transmission using wireless technologies.*

Resumo. *Com a evolução da computação móvel, dispositivos portáteis têm surgido no mercado com possibilidade de conexão a redes IP através de meios sem fio. A migração de terminais de acesso fixo para dispositivos móveis com acesso sem fio e a crescente facilidade para disponibilizar informações nesses dispositivos impõem novos desafios no desenvolvimento de aplicações. A segurança e a portabilidade são exemplos desses desafios que constituem a motivação desse trabalho. O objetivo principal é a introdução de um protocolo, chamado MobiS, para o desenvolvimento de aplicações seguras para transmissão de dados utilizando meios sem fio.*

1. Introdução

Uma enorme diversidade de dispositivos móveis tem surgido no mercado com possibilidade de conexão a redes IP utilizando meios sem fio [6], por exemplo, celulares com suporte a WAP, celulares GPRS, celulares i-Mode (no Japão), notebooks, Palms e Pocket PCs com Bluetooth e IEEE 802.11.

A possibilidade de mudança de terminais de acesso fixo para dispositivos móveis impõe novos desafios no desenvolvimento de aplicações [15]. A capacidade de mobilidade e de armazenamento, o baixo poder de processamento, o tamanho compacto e a possibilidade de comunicação sem fio com outros dispositivos e computadores conectados à Internet são exemplos desses desafios. Além disso, a utilização de meios sem fio para comunicação e a facilidade crescente para disponibilizar informações acarretam sérios riscos à segurança. Esses dispositivos oferecem mecanismos de segurança muito aquém dos oferecidos nas redes fixas.

O foco principal deste artigo é apresentar o protocolo MobiS, que permite o desenvolvimento de aplicações seguras para dispositivos móveis. Como estudo de caso descrevemos a construção da aplicação MobileMulta, que realiza a transmissão de infrações de multa de forma segura, do dispositivo móvel a um servidor de aplicação localizado na Internet.

O resto do artigo está dividido como segue. Na Seção 2 apresentamos um cenário genérico onde estão inseridas as aplicações para dispositivos móveis. Na Seção 3 apresentamos os riscos e as estratégias de segurança existentes a fim de enfatizar a motivação deste trabalho. Na Seção 4 introduzimos o protocolo MobiS para o desenvolvimento de aplicações seguras para dispositivos móveis. Na Seção 5 apresentamos as plataformas Java para o desenvolvimento de aplicações para dispositivos móveis. Na Seção 6 descrevemos o desenvolvimento de um estudo de caso, o MobileMulta, onde aplicamos o protocolo MobiS. Finalmente, na Seção 7 apresentamos as conclusões e os direcionamentos para trabalhos futuros.

2. Aplicações para dispositivos móveis

Aplicações corporativas desenvolvidas para dispositivos móveis normalmente estão inseridas em um cenário semelhante ao ilustrado na Figura 1:

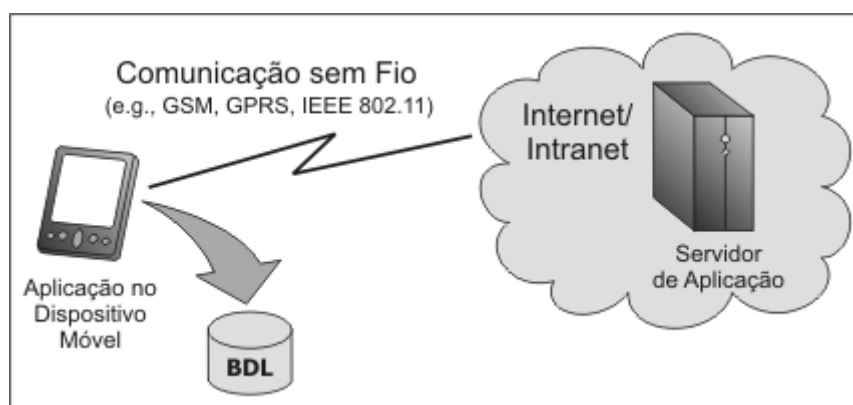


Figura 1. Um cenário genérico para dispositivos móveis transmitindo informações a um servidor de aplicação.

Neste cenário estão sendo levadas em consideração as aplicações nos dispositivos móveis, a comunicação sem fio e o servidor de aplicação. Essas aplicações, em geral, são executadas em modo *off-line* permitindo a inserção e manipulação de dados armazenados em uma base de dados local (BDL).

Em algum momento, a aplicação estabelece uma conexão utilizando uma tecnologia de comunicação sem fio onde realiza a transmissão dos dados ao servidor de aplicação. A periodicidade em que a conexão é estabelecida dependerá das limitações de memória impostas pelo dispositivo e das regras de negócio da aplicação (e.g., sincronizar os dados com o servidor após realizar a venda ao cliente).

A conexão pode ser do tipo: infravermelho com um telefone celular; discada utilizando modem e telefone; direta sem fio, através de uma rede local *wireless* (e.g., utilizando IEEE 802.11b ou *Bluetooth*) e direta com fio, através de um cabo serial de dados conectado a um computador. O cenário focaliza na comunicação sem fio, o que descarta alguns tipos de conexão citados acima.

Após o estabelecimento da conexão, a aplicação no dispositivo móvel transmite os dados armazenados na BDL ao servidor de aplicação que, ao recebê-los, oferece o tratamento definido nas regras de negócio. Ao receber a confirmação do servidor de que as informações transmitidas foram recebidas com sucesso, a aplicação apaga os registros da BDL liberando memória. A conexão é então desfeita pela aplicação retornando ao modo *off-line*.

3. Riscos e Soluções de Segurança

Este artigo utiliza o cenário genérico da Seção 2 para investigar os riscos existentes e as possíveis soluções de segurança que são apresentados na próxima seção. É necessário proteger o acesso à aplicação nos dispositivos móveis, a comunicação sem fio bem como o servidor de aplicação. Nas subseções seguintes apresentamos os riscos existentes e as possíveis soluções para este cenário.

3.1. Acesso indevido à aplicação nos dispositivos móveis

O extravio do dispositivo móvel, mesmo que temporário, pode acarretar o acesso indevido à aplicação bem como às informações armazenadas localmente no dispositivo. Na ocorrência desse acesso será possível a captura de informações, muitas vezes sigilosas, armazenadas em uma base de dados local ou em bancos remotos.

Os dispositivos móveis, geralmente, possuem senha de entrada que os protege contra acessos indevidos. Os dispositivos baseados no sistema operacional Windows CE, conhecidos como Pocket PC [16], são protegidos por senha de entrada a qual é mantida em hardware [16]. Esta proteção não é muito segura, pois o tamanho da senha e o conjunto de caracteres aceitos como entrada não são grandes o suficiente (e.g., 6 dígitos numéricos) para inviabilizar ataques de força bruta.

Nos dispositivos baseados no sistema operacional Palm OS [21], o mecanismo de senha é realizado em software e, portanto, pode ser burlado como apresentado em [1]. Existem algumas soluções comerciais para esse sistema operacional, como o OnlyMe [20], que bloqueia o dispositivo automaticamente após alguns segundos sem utilização. Outra solução disponível é o Sign-On [5], que só permite acesso após a autenticação através da assinatura e senha informadas pelo usuário.

Uma solução de segurança completa contra o acesso indevido à aplicação nos dispositivos móveis deve combinar a proteção contra o acesso indevido ao dispositivo com a proteção da aplicação através de senha de acesso.

3.2. Comunicação sem fio

Os meios de comunicação sem fio, além de possuírem limitações de largura de banda, estão sujeitos a variações de sinal irregulares (e.g., interferências externas). Outro problema relacionado ao meio sem fio está ligado à dificuldade de protegê-lo contra ataques passivos e ativos. Os dados transmitidos pelos dispositivos móveis podem ser interceptados, através da escuta do canal de comunicação, antes de chegar ao servidor de aplicação. A interceptação é um ataque passivo que põe em risco a confidencialidade das informações [4]. As informações interceptadas podem ainda ser modificadas e, posteriormente, retransmitidas, o que caracteriza um ataque ativo à integridade dos dados.

Outro ataque ativo está ligado à possibilidade de fabricação de informações. Um atacante pode fabricar informações e enviá-las ao servidor de aplicação. O servidor, ao receber as informações transmitidas pelo atacante, não será capaz de identificar a autenticidade da mensagem. Este é um ataque à autenticidade.

Na Figura 1, a conexão pode ser estabelecida utilizando o padrão IEEE 802.11b. Este padrão utiliza o protocolo WEP (*Wired Equivalent Privacy*) para provê proteção contra escutas e a outros tipos de ataques. Entretanto, neste protocolo já foram identificadas e reportadas falhas de segurança [4], tornando-o inseguro.

A conexão do dispositivo móvel pode ainda ser estabelecida utilizando os sistemas de telefonia celular. Por exemplo, pode-se utilizar o sistema GSM (*Global System for Mobile Communications*) ou o serviço de dados GPRS (*General Packet Radio Service*) para a transmissão de dados. No sistema GSM, a confidencialidade dos dados é garantida através do algoritmo *stream cipher* A5. Já no GPRS, o algoritmo de mesmo propósito utilizado é o GPRS/A5 – ou *GPRS Encryption Algorithm* (GEA) [7]. Ambos, A5 e GPRS/A5, são algoritmos proprietários, portanto, secretos. O algoritmo A5 já foi descoberto por técnicas de engenharia reversa tornando-o inseguro [9]. O algoritmo GPRS/A5 continua secreto.

Devido aos problemas de segurança existentes nos mecanismos de proteção nativos ao padrão IEEE 802.11b e aos sistemas de telefonia celular GSM e GPRS citados acima, mecanismos adicionais se fazem necessários para assegurar a comunicação. Dentre os mecanismos adicionais de segurança possíveis de serem implementados no cenário da Figura 1 podemos destacar VPN, o protocolo SSL/TLS, o protocolo KSSL e a criptografia na camada de aplicação, detalhados a seguir.

A segurança na transmissão pode ser garantida através de um túnel VPN, que é estabelecido entre o dispositivo móvel e o servidor de aplicação garantindo a confidencialidade das informações que trafegam por esse túnel. Entretanto, o uso de VPN denigre o desempenho da aplicação, aumenta o número de mensagens trocadas e sua utilização não é suportada por alguns dispositivos móveis devido a limitações de hardware [19].

A aplicação pode utilizar o SSL/TLS (*Secure Socket Layer/Transport Layer Security*) [28] que provê um canal seguro de transmissão entre os dispositivos móveis e o servidor. Entretanto, o SSL requer poder de processamento e memória nem sempre disponíveis nesses dispositivos [13]. Uma alternativa ao SSL/TLS seria o KSSL [29], uma versão do SSL de menor peso computacional implementado pela Sun®. O KSSL assume a presença do protocolo TCP, o que nem sempre será verdade em aplicações para dispositivos móveis.

Além das soluções citadas anteriormente, podemos utilizar os protocolos de criptografia desenvolvidos especificamente para redes fixas, implementando-os na camada de aplicação. No entanto, as limitações impostas pelos dispositivos impedem uma simples migração desses protocolos criptográficos. Considerando essas limitações e a segurança exigida pelas aplicações, são necessárias modificações adaptativas nesses protocolos de forma a permitir a sua utilização por esses dispositivos. Portanto, através da criptografia na camada de aplicação, obtemos uma proteção independente da infraestrutura de comunicação sem fio utilizada (e.g., IEEE 802.11, GSM, *Bluetooth*) pelos dispositivos móveis.

3.3. Servidor de Aplicação

O servidor de aplicação pode estar conectado à Internet estando exposto a ataques externos que podem causar interrupção de serviços, os chamados ataques de DoS (*Denial of Service* [22]). Esse é um ataque à disponibilidade. Neste caso, a implementação de um *Firewall* [18] pode protegê-lo contra determinados ataques.

4. Protocolo MobiS: Desenvolvendo Aplicações Seguras para Dispositivos Móveis

Na Seção 2, apresentamos o cenário em que estão inseridas as aplicações para dispositivos móveis consideradas neste trabalho. Na seção seguinte, foram apresentados os riscos existentes neste cenário e as possíveis soluções de segurança. Apresentaremos agora uma arquitetura para aplicações seguras aos riscos existentes no cenário em questão, conforme ilustrado na Figura 2, e o protocolo MobiS nas próximas subseções.

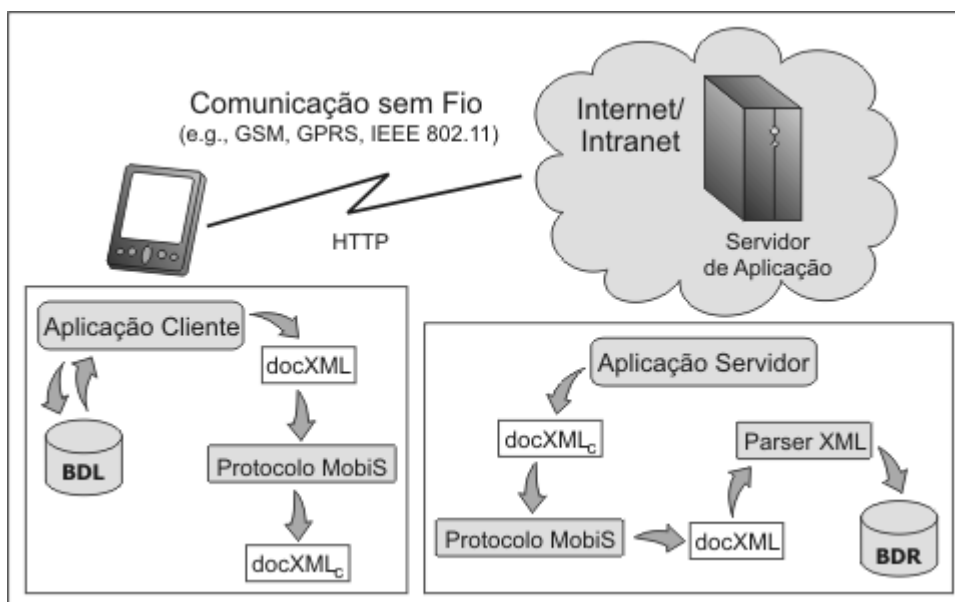


Figura 2. Uma Arquitetura para aplicações seguras

4.1. Solucionando os problemas de segurança

Nos dispositivos móveis, conforme apresentado na Figura 2, a aplicação cliente tem acesso a base de dados local (BDL), adicionando e manipulando informações. No momento em que o usuário deseja enviar as informações cadastradas na BDL, a aplicação cria um documento XML (*Extensible Markup Language*) contendo todo o seu conteúdo. O documento XML (docXML) é criado para facilitar a troca de dados entre as partes comunicantes. Ele é então cifrado utilizando o protocolo MobiS Simétrico (Mobilidade e Segurança) que será descrito posteriormente. A transmissão do docXML_c é realizada via HTTP (*HiperText Tranfer Protocol*).

O servidor, ao receber o docXML_c, decifra a mensagem utilizando o protocolo MobiS Simétrico verificando a autenticidade e a integridade. Caso a verificação seja realizada com sucesso, o *Parser XML* é instanciado e os dados recuperados do docXML serão inseridos no banco de dados remoto (BDR).

Para resolver o problema reportado na subseção 3.1, propomos o acesso à aplicação protegido pelo par de entrada login e senha. Na instalação da aplicação, o login e o resumo da senha do usuário são armazenados no dispositivo. O resumo é calculado utilizando uma função *hash*, tornando ainda mais seguro. Existem falhas de segurança reportadas em algumas funções *hash*, por exemplo, em [14] são apresentadas falhas no MD4. Portanto, a escolha da função *hash* a ser utilizada deve ser criteriosa, levando-se em conta o desempenho e segurança do algoritmo.

O mecanismo de segurança dos dados armazenados na BDL a ser adotado depende do recurso de armazenamento utilizado pela aplicação. Dependendo do dispositivo, do sistema operacional e da plataforma de desenvolvimento, a aplicação pode ou não ter acesso a arquivos de dados do sistema operacional (e.g., *pdb* no Palm OS [21]) ou acesso a clientes móveis de banco de dados. A segurança na BDL, portanto, é dependente da aplicação, como será vista na implementação do estudo de caso.

A segurança na transmissão de dados das aplicações, utilizando uma tecnologia de comunicação sem fio, pode ser alcançada de diversas formas conforme descrito na subseção 3.2. A utilização de VPNs não é adequada quando se deseja garantir a portabilidade da aplicação, já que em alguns dispositivos existem limitações (e.g., poder de processamento) impostas pelo *hardware* que inviabilizam sua utilização. Na utilização do SSL/TLS acontece o mesmo problema, conforme descrito na subseção 3.2. Já o KSSL apresenta desempenho aceitável, mas assume suporte ao protocolo TCP no dispositivo, o que nem sempre acontece.

Propomos então a modelagem e implementação de um protocolo de segurança na camada de aplicação, tornando-o independente da infra-estrutura de comunicação utilizada. Este protocolo, chamado MobiS, assegura a confidencialidade, integridade e autenticidade das informações transmitidas. O protocolo MobiS é baseado em protocolos existentes onde foram realizadas adaptações levando-se em conta as limitações de hardware dos dispositivos e a possibilidade de garantir maior portabilidade para a aplicação. São eles: o WEP, o SSL e o PGP (*Pretty Good Privacy* [3]). Descreveremos o protocolo MobiS a seguir.

O protocolo MobiS possui 3 (três) versões de modelagem: o MobiS Simétrico, o MobiS Assimétrico e o MobiS Híbrido. As denominações das versões do protocolo são referentes ao tipo de algoritmo de criptográfico utilizado, onde o MobiS Simétrico utiliza algoritmos de criptografia simétrica, o MobiS Assimétrico algoritmos de criptografia assimétrica e, por último, o MobiS Híbrido, que combina a utilização dos dois tipos de algoritmos. Até a finalização deste artigo, foi implementada apenas a versão MobiS Simétrico. As duas outras versões foram apenas modeladas, ficando a implementação e testes como trabalhos futuros. O estudo de caso apresentado na seção 6 utiliza o protocolo na versão MobiS Simétrico que apresentaremos na próxima subseção.

4.2. O Protocolo MobiS Simétrico

Este protocolo é baseado no protocolo WEP, utilizado em redes IEEE 802.11b para prover a confidencialidade dos dados. Apesar das falhas existentes, o protocolo WEP apresenta um esquema de segurança forte. Conforme mencionado em [4], as falhas do protocolo estão relacionadas ao algoritmo de criptografia RC4 e ao tamanho da chave utilizada.

O protocolo MobiS Simétrico é ilustrado na Figura 3. Em I descrevemos as operações realizadas pelo protocolo antes do envio da mensagem. Já em II descrevemos as operações realizadas pelo protocolo ao recebê-la. Detalharemos os 2 (dois) processos nas subseções seguintes.

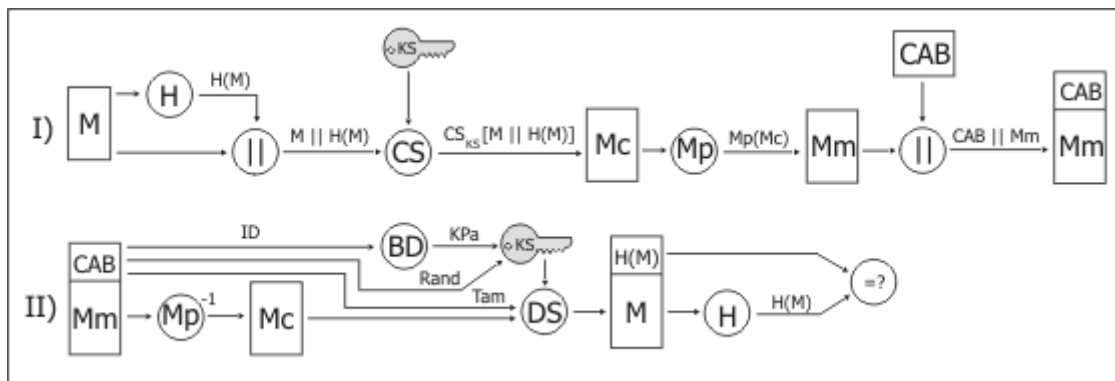


Figura 3. Protocolo MobiS Simétrico

4.2.1. Processo de Envio

A partir da mensagem M a ser enviada é calculado um resumo, $H(M)$, utilizando a função *hash* H . O resumo é então concatenado à mensagem gerando o texto resultante a ser cifrado, ou seja, $M || H(M)$. O texto resultante é cifrado utilizando um algoritmo de criptografia simétrica *block cipher* CS (e.g., *Triple DES*) utilizando a chave de sessão KS , assim, $Mc = CS_{KS}[M || H(M)]$. A chave de sessão pode ser gerada das seguintes formas: concatenando a chave simétrica KPa presente no dispositivo com um $Rand$ calculado a cada transmissão ($KS = KPa || Rand$) ou executando uma operação *Xor* (ou exclusivo) entre KPa e o $Rand$ gerado ($KS = KPa \oplus Rand$).

Depois de cifrar a mensagem, é aplicado sobre o Mc um algoritmo de conversão Mp (e.g., *Base64*) transformando caracteres inválidos gerados na criptografia em caracteres aceitáveis para a transmissão ($Mm = Mp(Mc)$). Este processo de mapeamento deve ser realizado pois os algoritmos de criptografias geram caracteres que não são transmitidos corretamente por alguns protocolos como o *MIME* e o *HTTP*. Algoritmos de mapeamento transformam bytes (256 possíveis caracteres) em uma tabela de caracteres menor. O *Base64* é um desses algoritmos, que transforma byte em caracteres alfanuméricos de acordo com uma tabela usando apenas 64 caracteres. A mensagem final aumenta em torno de 30%.

Ao Mm é adicionado um cabeçalho, CAB , contendo os campos: tamanho da mensagem original (TAM), o identificador do dispositivo (ID), o identificador de combinações de algoritmos (ICA) (e.g., um identificador para a combinação *Triple DES* + *MD5* + *HexEncoder*) e o $Rand$ gerado.

O envio do campo TAM no CAB é necessário, pois os algoritmos de criptografia simétrica *block cipher* requerem que o tamanho da entrada seja múltiplo do tamanho do bloco que o algoritmo trabalha (e.g., o *Rijndael* trabalha com blocos de 16 bytes). Sendo assim, é realizado antes da criptografia um *padding* ao final da mensagem para tornar seu tamanho múltiplo do tamanho do bloco do algoritmo.

4.2.2. Processo de recebimento

No recebimento da mensagem, o CAB é lido e o processo inverso de mapeamento (Mp^{-1}) é realizado sobre o restante da mensagem. Através do ID, é possível recuperar a chave simétrica KPa no banco de dados de chaves (BD). A chave KPa é concatenada (ou realizado um Xor como explicado anteriormente) com o *Rand* enviado, gerando a chave KS que será utilizada para decifrar a mensagem (DS). A função *hash* é aplicada sobre os dados decifrados, gerando um resumo que será comparado ao resumo que foi enviado. Se forem iguais, a integridade da mensagem foi assegurada.

A robustez do protocolo MobiS Simétrico está diretamente relacionada à combinação dos algoritmos bem como do tamanho da chave de criptografia.

Um dos problemas presentes na criptografia simétrica é a vulnerabilidade a ataques de texto cifrado conhecido (*known cipher text*). No MobiS Simétrico este problema é resolvido através do uso do número randômico *Rand* na composição da chave de sessão KS.

O MobiS Simétrico garante a confidencialidade dos dados, pois somente as partes comunicantes (dispositivo móvel e servidor) podem recuperar os dados originais. A integridade dos dados é assegurada através da função *hash*. Caso os dados sejam alterados, o resumo calculado no recebimento não coincidirá com o resumo enviado.

A fabricação de dados é evitada através da criptografia simétrica, pois somente quem possui a chave poderá produzir uma mensagem válida. Ataques de *reply* podem ser realizados, o que deve ser resolvido pela aplicação. Uma possível solução seria adicionar um código incremental que identifica unicamente os dados enviados. Assim, o servidor poderá descartar os dados recebidos com o mesmo código.

Outro problema existente na utilização da criptografia simétrica está relacionado à distribuição de chaves. O protocolo MobiS não se preocupa com essa questão, ficando a cargo do engenheiro de software escolher um dos mecanismos de distribuição de chaves existentes. No estudo de caso, detalharemos o mecanismo de distribuição de chaves utilizado.

5. Plataformas Java Disponíveis para Dispositivos Móveis

Ao desenvolver aplicações para dispositivos móveis, uma das principais decisões a ser tomada se refere à escolha da plataforma de desenvolvimento a ser utilizada. Existem diversas plataformas disponíveis no mercado [12][25][27], baseadas em diferentes linguagens de programação. O engenheiro de software deve considerar critérios que permitam a escolha da plataforma de desenvolvimento mais adequada ao tipo de aplicação. Critérios como características do dispositivo, tamanho e desempenho do aplicativo gerado, facilidade de desenvolvimento e custo devem ser considerados.

Apesar das aplicações desenvolvidas em C possuírem alto desempenho, ela é uma linguagem de difícil aprendizado e oferece pouca portabilidade às aplicações. A linguagem Java oferece às aplicações a portabilidade necessária, além de possuir uma curva de aprendizagem menor.

Dentre as plataformas existentes para o desenvolvimento de aplicações Java para dispositivos móveis, destacamos: a J2ME (*Java 2 Micro Edition*) e o SuperWaba. Nas

próximas subseções descrevemos as principais características dessas plataformas, além de apresentarmos uma comparação entre elas.

5.1. SuperWaba

SuperWaba, assim como J2ME, implementa a máquina virtual Java adaptada para dispositivos móveis [25]. Atualmente na versão 4.01 [25], foi desenvolvida por Guilherme Campos Hazan sendo uma evolução da plataforma Waba (criada por Rick Wild).

As primeiras versões do SuperWaba acrescentaram métodos nativos e novas classes ao Waba (e.g., janelas *pop-up*). Em seguida, o código foi expandido para dar suporte ao sistema Windows CE. A partir daí o SuperWaba tornou-se uma plataforma de desenvolvimento e não apenas uma extensão do Waba, adicionando novos componentes gráficos, tipos de dados *float* e *double*, suporte a bibliotecas C e Java, *ditherização* (exibição de cores que não estão no conjunto padrão de cores) automática de imagens, funcionalidades para a construção de jogos, janelas *pop-up* arrastáveis, *threads*, e, por final, alto desempenho alcançado através da otimização do interpretador. Oferece também bibliotecas de acesso a periféricos, como GPS e impressoras fiscais.

As APIs disponíveis no SuperWaba são [25]: *java.lang* (as aplicações importam implicitamente), *waba.fx* (inclui a classe *Graphics*), *waba.io* (entrada/saída), *waba.sys* (classes de interface com o sistema operacional), *waba.ui* (interface com o usuário), *waba.util* (classes utilitárias) e *superwaba.ext* (classes para o acesso a periféricos). As aplicações desenvolvidas em SuperWaba apresentam portabilidade com os sistemas operacionais Palm OS e Windows CE, não apresentando suporte aos sistemas dos celulares. A plataforma J2ME apresenta portabilidade com telefones celulares, como veremos a seguir.

5.2. J2ME

O J2ME é um conjunto de tecnologias e especificações projetadas pela *Sun Microsystems* [23], que visam o desenvolvimento de aplicações para dispositivos de pequeno porte (geralmente com suporte à comunicação sem fio). As especificações da plataforma são divididas em dois grupos: as configurações (*configurations*) e os perfis (*profiles*) [12].

As configurações especificam a máquina virtual Java e o conjunto básico de APIs para um determinado grupo de dispositivos. A configuração CLCD (*Connected, Limited Device Configuration*) é destinada a dispositivos pequenos, tais como PDAs e celulares, com memória de 512K ou menos e com conexão com a rede instável. O perfil para essa configuração é o MIDP (*Mobile Information Device Profile*). A especificação do MIDP encontra-se na versão 2.0 [10], mas ainda não existem dispositivos que suportam esta versão, sendo mais utilizada na versão 1.0 [12].

A máquina virtual Java para o CLDC/MIDP é a KVM (*Kilobyte Virtual Machine*). A Figura 4 ilustra o relacionamento entre as máquinas virtuais da plataforma Java (e.g., J2EE, J2SE) e a KVM.

A configuração CDC (*Connected Device Configuration*) é destinada a dispositivos de maior porte, que possuem 512K ou mais de memória, conexão de dados permanente e alta taxa de transmissão. Na Figura 4 observa-se que a CDC está sobre a

máquina virtual JVM, demandando maior capacidade de processamento e memória no dispositivo. Já a CLDC é implementada sobre a KVM, que possui um subconjunto de APIs da JVM projetada para executar em dispositivos com limitação de recursos. Desse conjunto de APIs destacamos: java.lang, java.io e java.util. Possui também uma API genérica de conexão, a javax.microedition.io.

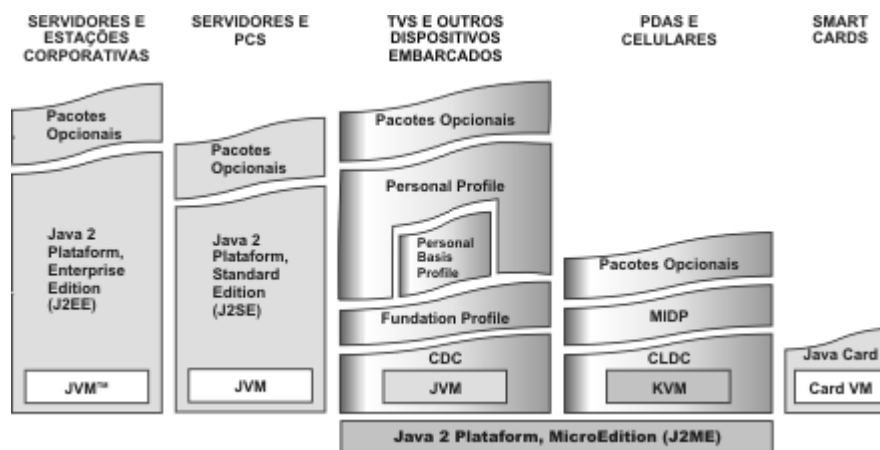


Figura 4. Plataforma Java Genérica e o J2ME (adaptada de [12])

A plataforma J2ME MIDP 1.0 não permite acesso a arquivos de sistema, ou seja, não é possível ter acesso a arquivos PDB (*palm database*) do Palm OS. A plataforma oferece como recurso de persistência de dados o RMS (*Record Management System*), consistindo de um conjunto de classes que realizam as operações comuns de banco de dados. As aplicações J2ME CLDC/MIDP são chamadas de MIDlets, que são empacotadas em arquivos JAR (*Java ARchive*) chamados de MIDletsSuites.

O MIDletSuite possui uma área de memória persistente reservada. Nesta área são armazenados, entre outras informações, os *Record Stores*, que são os arquivos de dados do RMS (similares às tabelas em um banco convencional). A KVM não permite que MIDlets pertencentes a outros MIDletSuites tenham acesso a esses *Record Stores*, ou seja, somente os MIDlets pertencentes ao MIDletSuite que os criou podem ter acesso aos dados [12][15][17].

A especificação do J2ME CLDC/MIDP 1.0 exige que as máquinas virtuais tenham o protocolo HTTP (*Hipertext Transfer Protocol*) implementado.

A plataforma J2ME permite o desenvolvimento de aplicações para dispositivos com os sistemas operacionais Palm OS e Windows CE, bem como apresenta portabilidade com alguns celulares (com suporte ao MIDP). Na próxima seção será apresentada uma comparação com as duas plataformas.

5.3. J2ME x SuperWaba

A grande vantagem da plataforma J2ME CLDC/MIDP 1.0 em relação ao SuperWaba é a portabilidade com diversos dispositivos móveis, incluindo celulares. Na tabela abaixo é apresentada uma comparação entre as duas plataformas, seguindo alguns critérios que definimos como importantes para a escolha da plataforma a ser utilizada no desenvolvimento das aplicações.

Tabela 1. Comparação entre J2ME CLDC/MIDP e SuperWaba

Critério	J2ME CLDC/MIDP (1.0)	SuperWaba
Portabilidade (Palm OS/ Windows CE)	SIM	SIM
Portabilidade com celulares	SIM	NÃO
Facilidade de Implementação	Boa	Boa
Tamanho da JVM para Palm OS	KVM da Sun: 619 KB	226 KB
Confiabilidade	Alta	Razoável
Custos	JVMs pagas para PDAs	Licença LGPL
Documentação disponível	Boa	Razoável, restrita ao <i>site</i> [25]
Acesso a arquivos do sistema	NÃO	SIM
Suporte a Sockets	NÃO (MIDP 2.0 Sim)	SIM
Suporte a acesso a IrDA	NÃO (MIDP 2.0 Sim)	SIM
Suporte a imagens	Formato PNG	Formato BMP
Controle do layout	Razoável	Alto
Suporte a https	NÃO (MIDP 2.0 Sim)	SIM

6. MobileMulta: um estudo de caso

O MobileMulta é uma aplicação desenvolvida para realizar o cadastro de infrações de trânsito. O usuário do sistema (i.e., o guarda de trânsito) utiliza um dispositivo móvel (e.g., Palm) para cadastrar as infrações na BDL. A transmissão dos dados ao servidor (i.e., o envio das infrações à central de trânsito) é realizada através de uma conexão via IrDA com um celular, o qual pode utilizar os sistemas GSM e GPRS.

O envio das infrações cadastradas na BDL ao servidor de aplicação exige a confidencialidade, integridade e autenticidade dos dados. Sendo assim, é necessária a utilização do protocolo MobiS Simétrico no MobileMulta como forma de garantir esses requisitos de segurança.

No protocolo MobiS Simétrico implementado no MobileMulta, utilizamos a combinação de algoritmos *block cipher* AES (*Advanced Encryption Standard*, algoritmo de criptografia baseado no Rijndael [2]) com chave de 128 bits, a função *hash* SHA1 (resumo de 160 bits) e o algoritmos de mapeamento Base64.

A chave de seção KS é gerada a partir da concatenação do KPa e o *Rand* ($KS = KPa \parallel Rand$). O KPa possui 96 bits, o *Rand* 32 bits, compondo os 128 bits necessários para forma a chave para o algoritmo AES. O uso do *Rand* compondo a chave KS inviabiliza as técnicas de criptoanálise de *known cipher text* como descrito na subseção 4.2.2 .

A distribuição de chaves ocorre da seguinte forma: na instalação do MobileMulta a chave KPa é gerada no servidor e sincronizada com o dispositivo através do cabo de dados. Uma cópia da KPa é armazenada no banco de chaves (BD) localizada no servidor, associada ao identificador do dispositivo ID. Portanto, a chave está protegida contra *eavesdropping*. A chave KPa é trocada a cada semana, período em que o guarda de transito comparece à central para a manutenção do sistema. Este período é determinado de acordo com as regras de negócio da aplicação, já que é possível ataques de força bruta como citado em [8].

Para o desenvolvimento do MobileMulta, utilizamos a plataforma J2ME CLDC/MIDP por apresentar portabilidade com alguns celulares (com suporte a MIDP 1.0) e com os sistemas Palm OS e Windows CE [11]. Utilizamos a IDE de

desenvolvimento *Sun One Studio* [24] integrada ao *Wireless ToolKit* 1.0 [12], que são ferramentas livres e amplamente utilizadas para o desenvolvimento em J2ME.

A plataforma J2ME CLDC/MIDP 1.0 não possui APIs para o tratamento de aspectos relacionados à segurança (e.g., suporte ao SSL). Um grupo de desenvolvimento, a *Legion of the Bouncy Castle* [26], disponibiliza, livremente, APIs J2ME que implementam os algoritmos criptográficos mais conhecidos. Utilizamos, portanto, as implementações do AES, SHA1 e Base64 desse grupo para compor o protocolo MobiS Simétrico.

O armazenamento dos dados no dispositivo é realizado utilizando *Record Stores* RMS. Dessa forma, garantimos a segurança dos dados, já que a máquina não permite o acesso através de outra aplicação, como descrito na seção 5.2.

Para realizar a transmissão dos dados utilizamos o protocolo HTTP. Dessa forma, garantimos a portabilidade da aplicação, já que todas as máquinas virtuais implementam este protocolo, como descrito na seção 5.2. A utilização de *sockets* também é possível, mas limitaria a utilização da aplicação a um grupo menor de dispositivos.

A Figura 5 ilustra os *screen shots* da aplicação *MobileMulta* implementada.

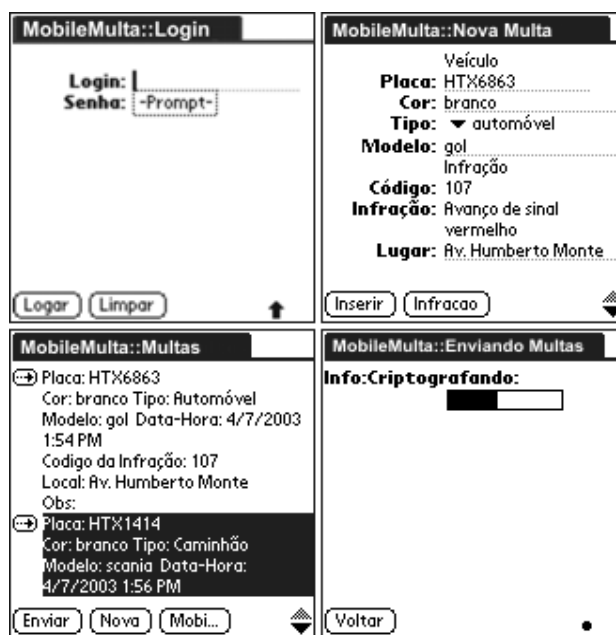


Figura 5. Screen shots do MobileMulta

O *MobileMulta* foi testado utilizando a máquina virtual da Sun nos Palms M130 e M515 (ambos com processadores de 33 Mhz) com 8MB e 16MB de memória, respectivamente. Utilizamos um celular Siemens S45 para permitir a comunicação com o servidor através de uma conexão discada GSM. Realizamos outro teste utilizando o celular Motorola A388 (2 Mb de memória, processador de 10 Mhz e suporte J2ME MIDP 1.0) utilizando o sistema GPRS para transmitir as infrações. Nos testes de simulação realizados, obtivemos um desempenho satisfatório. A utilização do protocolo MobiS Simétrico pela aplicação atrasou, em média, 250 ms por infração enviada.

7. Conclusão e Trabalhos Futuros

Este artigo apresenta o protocolo MobiS, que é uma solução para o desenvolvimento de aplicações seguras para dispositivos móveis. Além disso, apresentamos uma arquitetura para o desenvolvimento de aplicações seguras utilizando este protocolo. Para demonstrar a viabilidade da arquitetura, implementamos um estudo de caso, o MobileMulta, que utiliza protocolo MobiS Simétrico para prover a confidencialidade, integridade e autenticidade das infrações transmitidas.

Como trabalhos futuros, podemos citar: implementação e testes das versões Assimétrico e Híbrido do protocolo MobiS; avaliação de desempenho do protocolo MobiS Simétrico com outras combinações de algoritmos (e.g., TripleDES + MD5 + HexCodec) e testes da solução utilizando outros meios de transmissão (e.g., IEEE 802.11b).

8. Referências

- [1] @ STAKE. PalmOS Password Retrieval and Decoding. Disponível em <http://www.securiteam.com/securitynews/PalmOS_Password_Retrieval_and_Decoding.html>. Acesso em: 20 mar 2003.
- [2] AES Advanced Encryption Standard. Disponível em:<<http://csrc.nist.gov/CryptoToolkit/aes/>>. Acesso em: jan 2003.
- [3] An Open Specification for Pretty Good Privacy. Disponível em: <<http://www.ietf.org/html.charters/openpgp-charter.html>>. Acesso em: jan 2003.
- [4] Borisov, N.; Goldberg, I.; Wagner, D.; Intercepting Mobile Communications: The Insecurity of 802.11. p. 1-9, 2000.
- [5] The Power to Sign Online. Disponível em: <<http://www.cic.com>>. Acesso em: 11 mar. 2003.
- [6] Dornan, Andy, The Essential Guide to Wireless Communication Applications, Prentice Hall Inc., 2001. ISBN 0-13-031716-0.
- [7] ETSI TS 101 106. Digital Cellular Telecommunications System (Phase 2+); General Packet Radio Service (GPRS); GPRS ciphering algorithm requirements. European Telecommunications Standards Institute, mai 2001.
- [8] Gilmore, John. Cracking DES: Secrets of Encryption Research, Wiretap Politics & Chip Design. 1998. 272p
- [9] Golic, J. "Cryptanalysis of Alleged A5 Stream Cipher" EUROCRIP'97, LNCS 1233,239-255, Springer-Verlag
- [10] Guimarães, L. Sun e indústria wireless lançam MIDP 2.0. Sun Microsystems Portugal. Disponível em: <<http://pt.sun.com/noticias/imprensa/2002/021210.html>>. Acesso em: dez 2002.
- [11] Helal, S. Pervasive Java, Part II. Pervasive Computing, IEEE, vol1, p85 –89, Abr 2002.
- [12] J2ME – Java 2 Platform, Micro Edition. Apresenta a tecnologia J2ME. Disponível em <<http://java.sun.com/j2me>>. Acesso em: set 2002.

- [13] Kant, K.; Iyer, R.; Mohatappa, P. Architectural impact of secure socket layer on Internet servers. Computer Design. Anais do 2000 International Conference on Computer Design, 2000. Austin, TX, USA, p.7-14
- [14] Kasselmann, P.R. A fast attack on the MD4 hash function. Communications and Signal Processing (COMSIG97). Anais do South African Symposium, p.147-150, set. 1997.
- [15] Loureiro, Antônio; Minelli, André. Uma Ferramenta para Desenvolvimento de Aplicações para Dispositivos Móveis. Simpósio Brasileiro de Redes de Computadores, 20º, Búzios, v. 2, p. 571 - 586, 2002.
- [16] Microsoft Mobile. Tecnologia Microsoft para dispositivos móveis. Disponível em: <<http://www.microsoft.com/windowsmobile/default.mspx>>. Acesso em: 20 jun 2003.
- [17] Muchow, John W. Core J2ME. Sun Microsystems Press e Prentice Hall, 2001. 736p.
- [18] Murthy U.; Bukhres O.; Winn W.; Vanderdez E.; Firewalls for Security in Wireless Networks. P. 1-9, 2000.
- [19] Nunes, Bruno, A. A.; Moraes, Luís F. M. Avaliando a Sobrecarga Introduzida nas Redes 802.11 pelos Mecanismos de Segurança WEP e VPN/IPSec. Workshop de segurança - WSEG. In: Anais SBRC 2003.
- [20] OnlyMe. Tranzoa Home Page. Disponível em: <<http://www.tranzoa.com>>. Acesso em: 10 mar 2003.
- [21] Palm. Products, Services and Company Information. Informações de produtos e serviços para Palm. Disponível em: <<http://www.palm.com>>. Acesso em: jun 2003.
- [22] Stallings, William. Network security essentials: applications and standards / William Stallings. ISBN: 0-13-016093-8
- [23] Sun Microsystems. Forte for Java 4, Mobile Edition Getting Started Guide. Sun Microsystems. Disponível em: <http://forte.sun.com/ffj/documentation/ffjmetut.pdf>>. Acesso em: set 2002.
- [24] Sun[TM] One Studio 5, Standard Edition. Disponível em: <<http://www.sun.com/software/sundev/jde/buy/index.html>>. Acesso em: set 2002.
- [25] Superwaba – The Real Power of Mobile Computing. Disponível em: <<http://www.superwaba.com.br>>. Acesso em: set 2002.
- [26] The Legion of Bouncy Castle. Cryptography API para Java. Disponível em <www.bouncycastle.org> Acesso em: fev 2003.
- [27] Winkle W. V.; Palm Business Applications. The Ultimate guide to mobile computing: 2002 mobile Technology. p. 82-89, 2002.
- [28] RFC 2246. “The TLS Protocol version 1.0” Disponível em <<http://www.faqs.org/rfcs/rfc2246.html>>. Acesso em: 15 ago. 2003.
- [29] Gupta, V.; Gupta, S.; Experiments in Wireless Internet Security. Wireless Communications and Networking Conference – WCNC 2002. v. 2, p. 17-21, mar, 2002.