PEDRO ANCONA LOPEZ MINDLIN

μP: UMA SOLUÇÃO DE MICROPAGAMENTOS

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Mestre em Engenharia

São Paulo
2002

Para o meu amor, com amor.

Agradecimentos

Agradeço imensamente a todos os que me ajudaram durante a elaboração deste trabalho; agradecimentos especiais vão para:

- minha orientadora, Tereza Cristina Carvalho, pelo incentivo que me deu, principalmente na fase de conclusão do processo;
- o Professor Wilson Ruggiero, pelo entusiasmo com que sempre tratou o tema e por seu apoio fundamental;
- Christiane Schweitzer, por ter coordenado o projeto e pelo apoio à elaboração desta dissertação;
- Osvaldo Cabral Neto, pela inestimável ajuda na implementação do sistema.
- Ana Bia, minha namorada, pelo apoio, infinita paciência e pelos chocolates;
- minha mãe, Lilia, meu pai, Sérgio, e meus irmãos Paulo e Manoela por serem minha família e estarem sempre dispostos a me ajudar quando precisei;
- Oscar Vilcachagua, pela paciente ajuda com a formatação e problemas relacionados.
- meus colegas do LARC, especialmente Fernando, Vivian, Leo, Luís, Karin, Armin, Marcel e Leticia

Resumo

O desenvolvimento do comércio eletrônico brasileiro possibilita o surgimento de uma nova tendência: a venda de informações em formato digital; esse tipo de produto apresenta as características de envolver baixos valores e requerer transações rápidas, que são inviáveis para os sistemas tradicionais de pagamento eletrônico. Os micropagamentos surgem como alternativa nesse cenário, possibilitando a realização de transações eletrônicas rápidas e de baixo custo. Este trabalho apresenta um sistema de micropagamentos chamado μ**P**, que se caracteriza pela geração, em uma instituição central, de fichas eletrônicas que são compradas e distribuídas entre os clientes e facilmente verificadas — e portanto aceitas como pagamento — por vendedores atuantes no comércio eletrônico. O μ**P** se diferencia por utilizar apenas um conjunto de fichas que permitem realizar compras em qualquer vendedor cadastrado.

Abstract

*The development of electronic commerce in Brazil has led to a new trend: the distribution of digital information; this kind of product is characterized by involving low values and requiring fast transactions that are not suitable to existing electronic payment systems. Micropayment systems come as an alternative, allowing for the implementation of such transactions at low costs. This work introduces μ**P**, a micropayments system based on central generation of electronic coins that are bought by and distributed among clients and easily verified — and thus accepted as payment — by electronic commerce vendors. It differentiates itself by generating a single group of tokens that can be used for shopping among all of these vendors.*

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

O surgimento das redes de computadores, com a possibilidade de troca imediata de informações entre empresas e seus clientes e fornecedores, expandiu imediatamente as formas de realização de negócios, dando origem ao comércio eletrônico.

Nos países desenvolvidos, e especialmente nos EUA — onde surgiu todo o conjunto de tecnologias que o viabilizam, o comércio eletrônico se desenvolveu com maior velocidade, gerando grande movimentação tecnológica, acadêmica e econômica: muito se estudou, discutiu, desenvolveu e negociou a respeito de formas de realizar as transações de negócios de forma eletrônica. No início da década de 1990, com o surgimento e expansão da WWW (*World Wide Web*) houve um enorme crescimento na utilização da Internet pela população mundial, nunca antes visto em nenhum outro meio de comunicação. Muitas empresas passaram a vender seus produtos diretamente aos consumidores através da Internet.

Inicialmente, o crescimento do comércio eletrônico foi explosivo, pois os baixos custos de implantação e operação levaram ao surgimento de muitas pequenas empresas (conhecidas como *start-ups* ou *.com's*) dedicadas exclusivamente às vendas eletrônicas, sem existência física fora da Internet. Com o passar do tempo, no entanto, muitas dessas empresas faliram, pois seu modelo de negócio não se mostrou sustentável — a movimentação de vendas era baixa pelo simples motivo de que essas empresas não tinham como divulgar e firmar suas marcas de maneira sólida em um mundo totalmente virtual. Outro motivo para o fracasso de muitas empresas era o fato de que, após realizado facilmente o processo de venda eletrônica de um produto, era preciso entregá-lo fisicamente, o que se mostrou ser uma operação custosa e complicada.

Nessa "selva de pedra virtual" lentamente ficou clara a tendência de que sobreviveriam apenas as grandes empresas cuja marca já estivesse estabelecida e cuja infraestrutura permitisse a integração completa do processo de venda, incluindo entrega e serviços de pós-venda. Essa tendência se confirmou no final da década de 1990, quando

a "bolha" das empresas *.com* estourou, mostrando que muitas delas sobreviviam apenas de previsões otimistas de lucros futuros. Apenas as grandes restaram.

No Brasil o movimento em torno do comércio eletrônico não foi diferente do descrito acima, apenas em menor escala e ligeiramente atrasado, como ocorre nos países em desenvolvimento. Nosso comércio eletrônico evoluiu timidamente no início, espelhando-se na experiência norte-americana, e passou pelo mesmo processo de inchaço artificial experimentado pelo mercado daquele país. A queda aqui também foi brusca.

Os ataques terroristas de 11 de setembro de 2001 também tiveram um profundo impacto sobre o mercado de tecnologia da informação e, conseqüentemente, sobre o comércio eletrônico. O processo de globalização, infelizmente, não funciona como uma via de duas mãos equalitária; pelo contrário, é mais assemelhado a um *link ADSL*, em que o *download* tem banda maior que o *upload*: o desenvolvimento chega em passos lentos, mas a crise nos toma rapidamente — a recessão mundial afetou imediatamente o nosso mercado, criando um processo de retração do qual as empresas ainda não saíram.

Apesar de todas as dificuldades, no entanto, o desenvolvimento do comércio eletrônico não parou por um segundo desde o seu surgimento. Muita tecnologia e novos modelos de negócio estão continuamente sendo desenvolvidos. A razão por trás dessa aparente contradição são as vantagens que o comércio eletrônico oferece em relação ao comércio tradicional: a redução de custos e o contato direto entre a empresa e o cliente, algo que nem sempre é possível em razão da presença dos intermediários.

Embora seja utilizado por uma parcela ainda muito pequena da população (esse número, evidentemente, difere muito de país para país), o comércio eletrônico é uma forte aposta de muitas empresas. O mercado para esse tipo de atividade, embora pequeno, continua crescente nos países em desenvolvimento como o Brasil, e já está bem estabelecido nos Estados Unidos.

1.1 Motivação

O comércio eletrônico em sua forma mais simples — o varejo — funciona da seguinte maneira: o usuário acessa a página da empresa na Internet, seleciona o produto que quer comprar, paga por ele — o que inclui o frete, se o produto é físico — e o recebe no tempo

apropriado. Essa operação hoje corriqueira é aparentemente simples, mas envolve muita tecnologia: a empresa tem que construir a página em que vende o produto, integrá-la a um banco de dados de forma dinâmica, prover a integração do processo eletrônico de compra ao seu fluxo de caixa e ao sistema de logística, se for o caso, e prover *uma forma segura e adequada de pagamento* ao usuário, que por sua vez também precisa dispor de um computador e de *software* compatível com o sistema oferecido.

O foco desta dissertação está duas linhas acima: a forma de pagamento segura e adequada. O que é uma forma de pagamento adequada para o comércio eletrônico? Como se pode considerá-la segura?

A mais tradicional das formas de pagamento é o cartão de crédito. Nos Estados Unidos é um instrumento largamente aceito em qualquer instituição, seja ela de comércio eletrônico ou não. No Brasil, o uso do cartão de crédito está atrelado à estabilidade econômica: se a economia anda bem, o cartão de crédito é mais utilizado, embora os juros praticados pelas administradoras sejam exorbitantes; em tempos difíceis, porém, o uso desse instrumento fica muito restrito, pois a diferenciação de custos para os varejistas passa a ser muito grande, e os preços para compra com cartão de crédito são diferentes. De qualquer forma, o cartão de crédito é muito utilizado por permitir que a transação se complete imediatamente.

Outras formas comuns de pagamento no comércio eletrônico brasileiro são o cartão de débito e o boleto bancário. Ambas se baseiam na força e estruturação do sistema bancário deste país, que tem grande alcance e abrangência. Essas formas de pagamento, no entanto, dependem de autorizações bancárias que, via de regra, não são imediatas, o que atrapalha muito a transação, já que uma das características mais marcantes do comércio eletrônico é a velocidade.

Tendências

Montado o cenário, observemos agora quais são as tendências para desenvolvimentos futuros.

Estamos na era da informação. Essa frase já tão repetida e mastigada ainda merece a nossa atenção, pelo simples fato de representar uma verdade. Para o mundo industrializado, já ficou muito claro que o verdadeiro valor está na informação, no co-

nhecimento, e não nas coisas. Aquele que sabe fazer é capaz de reproduzir o que fez e se torna independente, enquanto que aquele que sabe apenas usar depende do que o outro sabe fazer. Assim se dá o processo de dominação tecnológica da globalização em todas as áreas; assim aconteceu o processo de transformação pelo qual tantas empresas passaram ao perceber que deviam valorizar seus funcionários em função do conhecimento de negócio que carregavam consigo.

No mundo da Internet, tudo foi transformado em informação: textos, música, vídeo, *software*, fotografia etc. A tendência natural da sociedade humana é a troca de bens e informações, e no mundo capitalista isso se dá através da troca de dinheiro em contrapartida. Uma das mais apostadas tendências para o futuro da Internet é a troca de pequenos pedaços de informação por pequenas quantias de dinheiro. E essa troca é diferente das trocas que se tem feito entre mercadorias e dinheiro através da rede, porque o seu custo é muito baixo — praticamente não há custos marginais envolvidos, somente o custo da própria informação que está sendo vendida. Imagine-se, nesse cenário, quanto pode custar um minuto de música, uma notícia de jornal, um artigo científico em texto integral ou uma imagem.

Essa troca de pequenas coisas pode parecer insignificante, ainda mais para ser a motivação de uma dissertação de mestrado em engenharia, mas não é. A engenharia existe para fazer com que as coisas funcionem, e trocar pequenas quantidades de informação por dinheiro através da Internet necessita de uma boa quantidade de engenharia, como veremos adiante.

O grande problema que se coloca é que o custo de uma transação de comércio eletrônico tradicional — da ordem de dezenas de centavos [10], ainda é maior que o preço de algumas informações que poderiam ser vendidas sob outras formas de pagamento. O objetivo deste trabalho é discutir uma proposta de forma de pagamento que permite a troca de pequenas quantidades de dinheiro — da ordem de centavos, de forma rápida e segura, para a obtenção de informações através da Internet.

1.2 Objetivos

O objetivo deste trabalho é estudar, planejar e implementar um sistema de micropagamentos eficiente e seguro. Esse sistema deve ser totalmente funcional e aplicável a casos práticos.

Foram estudados diversos sistemas anteriormente propostos e avaliadas as vantagens e desvantagens de cada um. Após esse estudo foi escolhido como base o sistema PayWord [18], por ser um sistema aberto e muito simples. A simplicidade também permitiu que fossem introduzidas modificações ao PayWord que atendessem aos nossos requisitos de eficiência, principalmente no que tange à latência, em função da tecnologia utilizada.

1.3 Organização

Esta dissertação está organizada da seguinte forma: no capítulo 2 o comércio eletrônico é discutido com maior profundidade, com uma exposição mais detalhada sobre as formas de pagamento mais comuns já existentes; no capítulo 3 são apresentados e discutidos trabalhos semelhantes relacionados à área de micropagamentos; no capítulo 4 apresentamos a arquitetura da solução proposta; no capítulo 5 é apresentada com detalhes a implementação experimental realizada; no capítulo 6 apresentamos os resultados obtidos e no capítulo 7 as conclusões e propostas futuras de trabalho.

Capítulo 2

Comércio Eletrônico

2.1 Introdução

Entende-se aqui por comércio eletrônico "a realização de toda a cadeia de valor dos processos de negócio em um ambiente eletrônico, por meio da aplicação intensa das tecnologias de comunicação e de informação, atendendo aos objetivos de negócio" [1], ou seja, comércio eletrônico é todo tipo de transação *on-line* com objetivo de desenvolver um negócio. Essa é uma definição abrangente que pretende abarcar todas as categorias de comércio eletrônico existentes, mas o foco principal desta dissertação é o comércio eletrônico visto como uma transação semelhante ao comércio tradicional: um consumidor compra um produto de uma empresa utilizando para tal alguma forma de pagamento.

Os produtos mais adequados para a venda *on-line* são aqueles para os quais o valor adicionado pelo processo eletrônico de compra é comparável ao valor do próprio produto: um CD ou livro, por exemplo, em oposição a uma casa, cujo alto valor não justifica a pequena economia gerada pelo comércio eletrônico. Isso nos leva a uma escala proposta em *The Economist* em 1997 [4], mostrada na figura 2.1. Ainda segundo Albertin [1], "a regra geral é que os consumidores *on-line* estão mais interessados em realizar compras com melhor informação e mais rapidamente do que necessariamente obter melhor preço". Isso deixa claro que, quanto mais se aproxima do nível de informação pura, distanciando-se do mundo material, mais adequada é a utilização do comércio eletrônico como base do modelo de negócio. Assim, a venda de informações puramente digitais como imagens, áudio e vídeo, cuja entrega ao comprador se dá através do mesmo meio que ele utiliza para a escolha do produto e para a realização do pagamento — o computador — é a que melhor se adequa ao comércio eletrônico.

Nossa transação de comércio eletrônico ficou agora reduzida a uma troca bilateral de informações: dinheiro para um lado e informação para o outro. No entanto, se

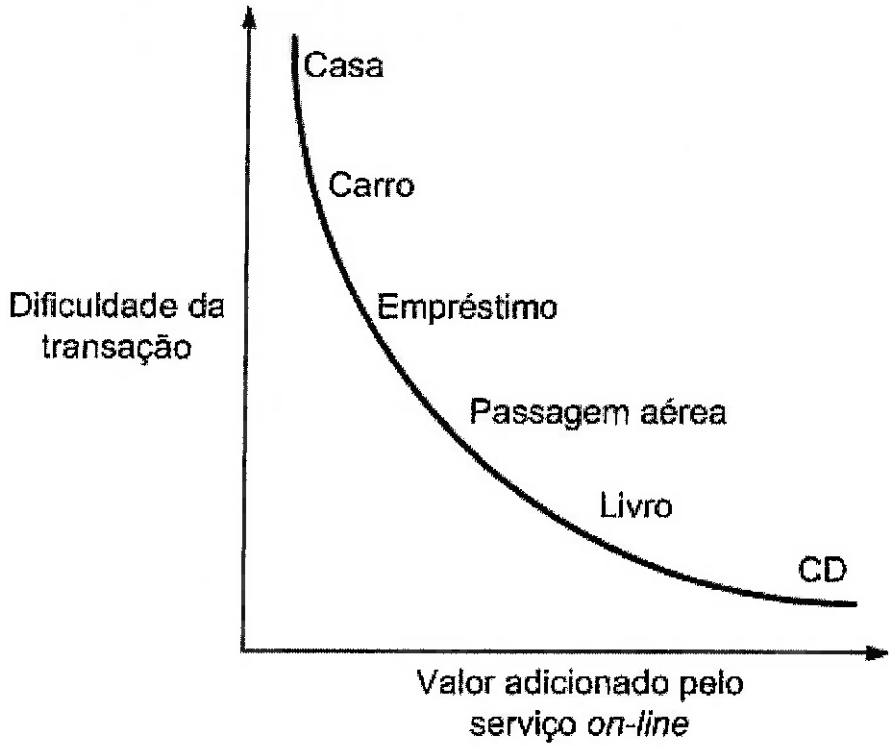


Figura 2.1: Níveis de vendas *on-line* [4]

pensarmos bem, a troca de dinheiro nada mais é que a troca de informação sobre o dinheiro. Esse ponto é muito importante: mesmo no sistema bancário, a compensação de pagamentos entre os bancos se dá através da troca de informações, uma vez que a troca física de dinheiro em grandes quantidades é muito arriscada e custosa. Chegamos assim a uma situação de troca bilateral de informações em que cada informação tem o seu formato: de um lado a informação é financeira e do outro tem a forma que se desejou adquirir.

A troca de informação financeira é o assunto principal deste trabalho. Essa troca, também chamada forma ou sistema de pagamento, pode acontecer de diversas maneiras, que serão vistas a seguir.

2.2 Sistemas de pagamento

No comércio tradicional brasileiro existem muitas formas de pagamento vigentes, algumas mais utilizadas que as outras. A taxa de utilização de algumas formas de pagamento está diretamente ligada a vários fatores: facilidade de uso, custo por transação,

segurança e outros. Algumas delas se aplicam fácil e diretamente ao comércio eletrônico, algumas precisam de adaptações e outras simplesmente não se aplicam, mas apresentam formas análogas.

dinheiro A primeira e mais comum forma de pagamento é o dinheiro. O dinheiro é a forma impressa ou cunhada em metal de uma garantia de valor assegurada pelo Estado, e é portanto de aceitação obrigatória em qualquer forma de transação financeira no país. É também a forma mais barata de pagamento, incorrendo em custo zero por transação. Embora possua a vantagem de ser universal e barato, o dinheiro tem alguns inconvenientes:

- sua propriedade é de seu portador, não havendo como estabelecer ou provar outra forma de propriedade que não a posse física — isso gera sérios problemas quanto à segurança de se portar dinheiro;
- é totalmente fungível, ou seja, não há diferença prática alguma entre cédulas do mesmo valor — isso também dificulta a comprovação de sua propriedade;
- o valor das cédulas é quantizado — 1, 2, 5, 10 etc. — e seu número é limitado[1], o que cria problemas para a troca de valores intermediários (troco).

A utilização do dinheiro no comércio eletrônico é impossível, pois o seu valor está associado intrinsecamente à cédula ou moeda. Não é possível transformar uma cédula ou uma moeda em informação, embora seja possível enviar informações sobre a posse das mesmas. A troca financeira feita em dinheiro, no entanto, deve ser realizada mecanicamente através do transporte físico das cédulas ou moedas.

A informação sobre posse de dinheiro pode ser chamada de dinheiro eletrônico. Essa informação, quando em formato eletrônico, pode trafegar tranqüilamente através de computadores e redes, sendo uma possibilidade realizável para o comércio eletrônico. A questão importante que surge sobre o dinheiro eletrônico, no entanto, é quanto à autenticidade da sua procedência. É preciso utilizar meios que comprovem com segurança

[1] para um panorama interessante e preciso acerca do dinheiro em circulação no Brasil, veja http://www.bcb.gov.br e clique em Informações Econômicas e Financeiras → Indicadores de Conjuntura → Dinheiro em Circulação

que aquela informação sobre a posse de uma quantia em dinheiro é verdadeira. Para isso é comum a utilização de cartões magnéticos e sistemas de criptografia protegidos por senhas. Um dos exemplos mais importantes de dinheiro eletrônico é o sistema Mondex[2], que conseguiu grande alcance internacional. A informação de crédito entre instituições bancárias para compensação de pagamentos também pode ser considerada dinheiro eletrônico.

cheque Outra forma de pagamento muito comum no Brasil é o cheque. Trata-se de uma ordem de pagamento garantida por uma instituição privada na qual o usuário mantém uma conta. O cheque é pessoal e intransferível, e só vale se assinado pelo seu titular, o que lhe confere um grau de segurança bem maior que o do dinheiro (embora os bancos não possam conferir todas as assinaturas de cheques de baixo valor). Outra vantagem do cheque é a possibilidade de ele ser preenchido com qualquer valor, o que facilita pagamentos de maior vulto ou de valores quebrados em relação às cédulas e moedas de dinheiro. O cheque, no entanto, também apresenta algumas desvantagens:

- seu custo é relativamente alto, restringindo seu uso a camadas sociais de maior poder aquisitivo e a pagamentos de maior valor[3]
- pode ser roubado e preenchido com valores quaisquer por terceiros, valendo-se do fato de que nem todas as assinaturas são conferidas.
- pode ser emitido sem a real existência de fundos que lhe garantam o pagamento. Como a verificação é feita *a posteriori*, os comerciantes incorrem no risco de arcar com o prejuízo.

O cheque também não pode ser utilizado diretamente no comércio eletrônico por sua natureza; assim como as cédulas e moedas de dinheiro, o cheque é feito de papel e não pode ser transmitido eletronicamente. Seu formato, entretanto, é facilmente transformado para a forma eletrônica, pois nada mais é que uma ordem de pagamento assinada. Se lembrarmos da existência das assinaturas feitas com certificados digitais essa transformação se revelará imediatamente.

[2] http://www.mondex.com

[3] geralmente os bancos cobram taxas extra de 1% para cheques abaixo de R$ 20,00

cartão de crédito Outra forma de pagamento muito utilizada é o cartão de crédito, que se baseia na garantia de pagamento da instituição administradora do cartão. Das três formas de pagamento discutidas até agora, é a única que se baseia em crédito ao invés de débito; a administradora não requer um depósito anterior ao gasto, e sim cobra posteriormente a conta dos gastos. O cartão de crédito, no entanto, é ainda mais caro que o cheque, e seu uso está limitado a uma parcela ainda menor da população.

O cartão de crédito não precisa existir fisicamente para a realização de uma compra. Através de métodos como a Transação Eletrônica Segura (SET ou *Secure Electronic Transaction* — veja adiante) ou de protocolos como o SSL é possível transmitir o número do cartão de crédito entre usuário, vendedor e administradora, completando a transação de forma *on-line*, o que viabiliza essa forma de pagamento para ser utilizada imediatamente no comércio eletrônico.

boleto bancário Existe ainda um último sistema de pagamento muito utilizado no Brasil, o boleto bancário. Esse é um sistema de cobrança bancária tradicional que foi adaptado ao comércio eletrônico de três formas diferentes:

1. o comprador pede o produto, espera pela chegada do boleto pelo correio, paga no banco e envia o comprovante ao vendedor ou espera que ele confirme o pagamento. Esse é o método mais lento e antigo, já praticamente não utilizado;

2. o comprador pede o produto, imprime o boleto, paga no banco e envia o comprovante ou aguarda a confirmação. Esse método é mais ágil, mas ainda depende do tempo necessário para ir ao banco e para a confirmação do pagamento;

3. o comprador pede o produto e paga o boleto através de um serviço bancário *on-line*. Esse é o método mais rápido, porém a confirmação de recebimento do pagamento pelo vendedor ainda pode demorar de forma proibitiva.

SET A SET [22] (*Secure Electronic Transaction* ou Transação Eletrônica Segura) é um padrão desenvolvido pelas empresas Visa e Mastercard para tornar seguras as transações com cartões de crédito ou débito através de redes abertas [22]. A SET é fortemente baseada em assinaturas digitais para prover segurança, confidencialidade e

atomicidade para a transação. Seu surgimento provocou um movimento importante em favor do comércio eletrônico, pois representou uma forte aposta de empresas muito grandes no mercado.

A SET tem, no entanto, duas desvantagens: a demora para conclusão da transação (embora seja feita *on-line*) e a inadequação para baixos valores, em razão de seu alto custo. Esses fatores são derivados do fato de que a SET foi desenvolvida para trabalhar com a rede legada de operações de cartão de crédito, adicionando a ela uma camada de segurança. Como em toda herança recebida de sistemas anteriores, o preço a ser pago é sempre um aumento de custo. Nesse caso, o alto custo se deve também ao alto custo computacional gerado pela necessidade de geração e verificação de várias assinaturas digitais para cada transação e ao alto custo de comunicação (latência) que decorre do fato de a autorização para a compra ser dada imediatamente pela operadora de cartão de crédito.

2.3 Micropagamentos

Os micropagamentos são uma nova categoria de sistema de pagamento, proposta inicialmente por volta de 1992 e que se beneficiou muito do desenvolvimento de novas aplicações para funções de *hash* como SHA (*Secure Hash Algorithm*) [12] e MD5 (*Message Digest 5*) [16] por volta de 1996, com as propostas de Rivest [18], Pedersen [14] e Anderson [3]. Essas propostas serão anàlisadas mais detalhadamente no capítulo 3.

Micropagamentos, como o próprio nome já indica, são pagamentos de valores muito pequenos. O problema que os sistemas tentam resolver é como fazer esses pagamentos a custos ainda mais baixos que os já baixos valores dos pagamentos que realizam. Os sistemas de micropagamentos são intrinsecamente eletrônicos, pois se destinam originalmente para uso apenas em comércio eletrônico. Assim, sua adaptação a esse meio é perfeita.

Um dos maiores problemas que esses sistemas têm que resolver, como conseqüência de sua natureza totalmente eletrônica, é a interface com o mundo financeiro real. A solução desse problema muitas vezes é deixada pelos autores para outros protocolos, como por exemplo o SET.

Os sistemas de micropagamentos envolvem normalmente três entidades: o usuário, que é aquele que quer realizar uma compra, normalmente uma pessoa física diante de um navegador da Internet; o vendedor, que é a empresa ou pessoa que está vendendo seu produto ou serviço através da rede e o agente, que é a instituição — normalmente uma instituição financeira — responsável pelo sistema, por emitir as fichas e/ou garantir o seu pagamento, por manter as contas de usuários e vendedores atualizadas, enfim, por manter tudo funcionando. Alguns sistemas apresentam também um quarto elemento, um agente aglutinador de pagamentos, que funciona como um atravessador para evitar os custos associados a baixos volumes de transações.

Os sistemas de micropagamentos são normalmente classificados em duas categorias: *on-line* e *off-line*.

Sistemas *on-line* são aqueles em que o agente participa diretamente da transação, autorizando a compra no mesmo momento em que ela é realizada. Esse tipo de sistema é mais seguro pois a possibilidade de ocorrência de fraudes é menor, uma vez que o agente pode verificar instantaneamente a autenticidade do pagamento que está sendo feito. Por outro lado, a necessidade de verificação de cada pagamento por parte de uma entidade central traz três desvantagens:

1. o atraso causado por essa verificação pode ser grande, tornando-se um fator de inviabilidade do sistema;

2. a carga computacional sobre a entidade central pode ser muito grande se o número de usuários do sistema crescer demasiadamente (o que é, na verdade, o objetivo das propostas — tornarem-se formas universais de pagamento, tão aceitas quanto dinheiro);

3. a entidade central passa a ser um ponto único de falha; em caso de algum problema nessa central o sistema todo ficará comprometido.

Os dois últimos problemas enumerados acima podem ser contornados com um maciço investimento em *hardware*, para a obtenção de um sistema altamente escalável e redundante. Nesse caso, apenas o problema da latência restaria — o que já é suficiente para que a concorrência dos sistemas *off-line* se torne marcante.

Sistemas *off-line* são aqueles em que o agente não participa diretamente de cada transação. Seu papel é, normalmente, restrito a uma fase inicial antes da realização das compras — confecção de fichas, assinatura de certificados que garantem crédito e outras funções inicializadoras — e ao posterior pagamento aos vendedores das fichas gastas e eventuais cobranças aos usuários que se fizerem necessárias. A verificação da autenticidade da transação é feita diretamente pelo vendedor, com posterior revalidação por parte do agente.

Esse tipo de sistema tem a vantagem de apresentar transações mais rápidas, pois a latência do envio das mensagens ao agente é eliminada. A desvantagem é que a descentralização — o objetivo principal desse tipo de sistema — possibilita uma maior ocorrência de fraudes.

A maioria dos sistemas de micropagamento funciona com base no princípio de que se pode gerar eletronicamente uma moeda de forma única. A moeda é, na verdade, uma mensagem composta por um valor principal e outros campos que, possivelmente, autenticam o valor principal. É importante ressaltar que o valor a que nos referimos aqui não é o valor financeiro da moeda, mas o valor eletrônico que tem a propriedade de ser único. Esse valor geralmente é escolhido como sendo um número muito grande, da ordem de 2^{160}, que é o tamanho médio da saída produzida por funções como SHA. Valores dessa ordem são importantes por dois motivos: são grandes o suficiente para que sejam custosamente forjados por ataques de força bruta e são pequenos o suficiente para permitir fácil manipulação para a verificação de sua autenticidade.

As fraudes em sistemas de micropagamentos podem acontecer de várias formas, e variam bastante com a forma específica de cada sistema. Alguns tipos de fraudes, contudo, são comuns: a criação de fichas falsas e o gasto múltiplo de uma ficha, principalmente em vendedores diferentes. As medidas tomadas para resolver esses problemas têm formas semelhantes: para impedir a criação de fichas falsas são empregadas funções de mão única e assinaturas digitais em diversas combinações na criação das fichas e para impedir o gasto múltiplo são empregados esquemas de gerenciamento de conta, normalmente através do agente, mesmo que de forma tardia.

Os sistemas de micropagamentos permitem o surgimento ou a viabilização

econômica de uma série de aplicações no comércio eletrônico. Citaremos algumas a título de exemplo, retiradas de [24], mas as possibilidades que se abrem são inúmeras:

1. acesso a publicações científicas: muitas vezes só é possível ler um artigo quando se compra uma publicação inteira ou, pior, quando se assina a publicação por um certo período. Através de micropagamentos poderia-se ter acesso a artigos individuais;

2. acesso a notícias individuais de jornais: para se ler uma notícia de jornal é preciso comprar o jornal todo. As publicações poderiam inaugurar um novo modelo de negócio em que estariam à venda apenas pedaços de informação. Como os preços poderiam ser muito baixos, os micropagamentos fariam-se necessários;

3. acesso a bancos de dados: empresas poderiam vender acessos individuais de dados armazenados em suas bases por baixos preços por acesso através de micropagamentos;

4. acesso a páginas da Internet: muitas páginas que contêm informações importantes poderiam contar com recursos adicionais para sua manutenção através da cobrança de centavos para visualização.

5. novas formas de propaganda: em um processo inverso, mantenedores de páginas poderiam pagar pequenas quantias a seus usuários para que acessassem suas páginas comerciais.

O projeto SCDSID (Sistema de Controle e Distribuição Segura de Imagens e Documentos) [21], do consórcio LARC-SCOPUS, prevê a distribuição segura de documentos com controle de acesso. Cada documento tem propriedades associadas que podem ser controladas separadamente: visualização, impressão e cópia. O controle dessas propriedades poderia fazer com que um usuário pudesse visualizar uma imagem sem no entanto poder imprimi-la ou copiá-la, ou poderia permitir um número reduzido de visualizações e/ou impressões. O controle desses parâmetros poderia se dar através de micropagamentos: ao escolher uma imagem, o usuário envia uma ficha para cada visualização que quer fazer da imagem, e mais cinco fichas para cada impressão que quiser fazer. Se quiser fazer uma cópia da imagem ele pode então enviar cem fichas, por exemplo.

O movimento em torno dos esquemas de micropagamento foi muito grande nos meados da década de 1990 nos Estados Unidos, quando os primeiros esquemas foram propostos e o comércio eletrônico estava se tornando uma realidade estável. Desde então muitas tentativas comerciais foram feitas, mas poucas sobreviveram a longo prazo. O recrudescimento da Internet e das empresas .com no início do século XXI fez com que essas idéias voltassem ao papel.

Nos últimos anos os micropagamentos têm voltado às pautas dos pesquisadores (ver [10]), e nós acreditamos que existem possibilidades de esses esquemas funcionarem no Brasil, uma vez que o nosso mercado para o comércio eletrônico se desenvolveu mais tarde e está agora em uma fase de maturidade suficiente para aceitar mudanças mais ousadas em seu meio de funcionamento.

No próximo capítulo apresentaremos algumas das propostas mais proeminentes de micropagamentos apresentadas nos últimos anos.

Capítulo 3

Trabalhos Relacionados

A pesquisa relacionada à área de micropagamentos surgiu durante a segunda metade da década de 1990, beneficiada pelo surgimento de algoritmos de mão única, também conhecidas como funções de *hash* como as funções SHA *Secure Hash Algorithm*) [12] e MD5 (*Message Digest 5*) [16]. Essas funções foram inicialmente utilizadas em esquemas de controle de acesso por Haller [5]. Os trabalhos apresentados neste capítulo podem ser comparados resumidamente através da tabela 3.1, apresentada na página 24.

3.1 Sistemas originalmente propostos

Millicent

O primeiro trabalho importante surgido na área de micropagamentos foi o Millicent [9]. Esse trabalho propõe o uso de uma mensagem chamada *scrip* ou título para a realização de compras de pequenos valores, tipicamente abaixo de 50 centavos de dólar.

O título é uma mensagem contendo:

- a identificação de seu proprietário,
- a identificação do vendedor associado,
- o seu valor,
- uma data de expiração,
- um número serial,
- uma série de propriedades que descrevem o proprietário e
- um certificado digital que autentica o conteúdo acima.

Para realizar uma compra, o cliente primeiro obtém um título de um agente. Esse título é usado para adquirir, do agente, títulos específicos para cada vendedor. Os

títulos são trocados entre agente e vendedor e depois repassados ao cliente. Uma vez de posse do título do vendedor, o cliente o utiliza para realizar a compra. O título é enviado para o vendedor, que deduz o valor da compra do valor do título e reenvia o título com outro valor. Dessa forma, a manutenção da conta do cliente junto ao vendedor é feita dentro do próprio título.

O Millicent se baseia em um modelo de segurança simplificado, justificado pelo fato de as transações e os títulos serem de valores muito baixos. Segundo os autores, os títulos devem ser tratados como moedas de baixo valor — daquelas que não nos importamos por perder ou manter uma contabilidade estrita. O roubo de títulos dessa natureza seria então algo não rentável, e o roubo de muitos títulos seria facilmente detectado e combatido. Essa filosofia é compartilhada por quase todos esquemas de micropagamentos, incluindo o μ**P**.

O Millicent inspirou muitos outros trabalhos, tais como o NetCard [3], o Tick Payments [14], o PayWord [18] e o iKP [7], que são semelhantes e contemporâneos.

NetCard

O NetCard [3] é uma proposta de micropagamentos baseada na utilização extensiva de funções de *hash* para a geração e utilização de moedas eletrônicas. Cada moeda é uma cadeia de 64 *bits*, encapsulada por informações extra a respeito de usuário, vendedor, banco, moeda real à qual a moeda eletrônica se refere e produto sendo comprado.

Os autores propõem quatro protocolos para a utilização dessas moedas.

O primeiro deles prevê a geração de moedas pelo banco, com recebimento de pacotes de 100 moedas pelo usuário por vez. Ao iniciar um processo de compra, o usuário cria uma cadeia de moedas através da função de *hash*, utilizando como entrada a i-ésima moeda e os dados da compra. A desvantagem desse protocolo é a geração inicial de todas as moedas pelo banco. Esse protocolo se parece com o μ**P** exatamente nesse aspecto. O μ**P**, no entanto, não necessita do cálculo de uma nova cadeia de valores de *hash* após a geração das moedas.

No segundo protocolo a geração das moedas é feita pelo usuário, assinada e enviada diretamente ao vendedor. Esse protocolo é semelhante ao μ**P** exceto pelo fato de que a função de *hash* é executada no usuário. O fato de a cadeia de valores ser única e

baseada no *hash* de um valor inicialmente aleatório, no entanto, torna as propostas muito semelhantes.

O terceiro e o quarto protocolos do NetCard são baseados na utilização do sistema SET, e portanto fogem muito do escopo deste trabalho.

Payword

O PayWord [18] é um esquema baseado em crédito em que o usuário recebe do agente um certificado digital contendo seu nome, endereço IP, chave pública e outras informações. De posse desse certificado o usuário pode criar e assinar seqüências de valores de *hash* específicas para cada vendedor.

A cada novo dia, quando o usuário acessa pela primeira vez o *site* do vendedor, ele cria uma nova seqüência de valores de *hash* e envia ao vendedor apenas o último desses valores, assinado digitalmente com o certificado que recebeu do agente. Para cada compra subseqüente o usuário envia ao vendedor o valor imediatamente anterior. Lembrando que a função de *hash* é uma função de mão única, sabemos que é impossível ao vendedor descobrir qual o próximo valor a ser enviado pelo usuário. Quando esse valor é recebido, basta aplicar sobre ele a função de *hash* e se chegará ao valor anterior, autenticando assim o pagamento.

Ao final de cada dia, o vendedor envia ao agente um relatório contendo os certificados recebidos de cada usuário e a última ficha gasta, a partir dos quais o agente pode fazer o pagamento que lhe é devido em um só lote.

A principal vantagem do PayWord é o fato de a autenticação de cada pagamento ser feita através da aplicação de uma função de *hash*, que é muito rápida[1]. Sua principal desvantagem está no fato de o usuário ter que gerar assinaturas digitais para cada novo vendedor que visitar a cada dia; essa carga pode ser muito pesada se o número de vendedores visitados for grande.

[1] Aproximadamente 100 vezes mais rápida que a verificação de uma assinatura RSA e 10.000 vezes mais rápida que a geração de uma assinatura RSA, segundo os autores.

MicroMint

O MicroMint[2] [18] é um esquema muito interessante em que o agente gera moedas eletrônicas que podem ser trocadas livremente entre usuários e vendedores, e são garantidas contra falsificação. A tecnologia por trás dessas moedas está em produzir colisões de funções de *hash*, ou seja, encontrar k valores que produzam a mesma saída quando fornecidos como entrada a uma dessas funções. É claro que, para encontrar esses valores, é preciso executar essa função em quantidade de força bruta, e isso só é possível, em um tempo prático, com grande investimento em *hardware*. Uma vez feito o investimento, no entanto, o custo por ficha decresce ao ponto de sua produção ser economicamente viável. A falsificação, porém, continua sendo muito cara e a verificação de autenticidade das fichas é muito simples, dependendo apenas de se aplicar k vezes a função de *hash*. Uma grande vantagem do MicroMint sobre outros esquemas é o fato de não envolver assinaturas digitais, que são computacionalmente custosas, em momento algum. Embora seja simples e interessante, esse esquema é impraticável em bases experimentais, fugindo muito do escopo deste trabalho.

Tick[3] Payments

O esquema proposto por Pedersen [14] tem como objetivo inicial fazer o pagamento eletrônico de serviços temporizados tais como impulsos telefônicos. O autor apresenta uma proposta teórica muito semelhante ao esquema proposto neste trabalho, baseado em funções de mão única. As fichas são geradas no usuário, mas o autor propõe o uso de *hardware* especial para o cálculo e armazenamento das fichas, o que foge do escopo deste trabalho. Embora seja apenas teórico, sua relevância está no fato de ser contemporaneamente pioneiro no uso de funções de mão única como forma segura de gerar moedas eletrônicas.

iKP

o esquema baseado no iKP [7], da IBM, é semelhante ao esquema proposto neste trabalho: o usuário computa fichas de micropagamento através de uma função de mão

[2] *Mint* denomina o lugar em que se forjam moedas.
[3] *tick* significa sinal, marca

única como MD5 ou SHA. A proposta prevê dois tipos de relacionamento entre usuários e vendedores: um tipo de relacionamento freqüente, em que o usuário faz micro-compras repetidamente no mesmo *site* e um tipo de relacionamento casual, em que o usuário faz compras ocasionalmente em um *site*.

Para o relacionamento freqüente os autores propõem que o usuário gere uma quantidade de fichas exclusivas para aquele vendedor e passe a enviar uma ficha a cada compra, de acordo com o necessário. O envio da primeira ficha, a partir da qual seriam verificadas as outras, seria feito através do protocolo *i*PK, desenvolvido pela IBM para transações seguras com cartões de crédito.

Para o relacionamento ocasional os autores propõem a intermediação de um agente, sendo que o usuário enviaria os pedidos de produtos diretamente ao agente que, por sua vez, intermediaria a micro-compra. Dessa forma, segundo os autores, o volume necessário de transações entre usuário e vendedor para que o sistema seja economicamente viável passa a ser semelhante àquele do relacionamento freqüente entre usuário e vendedor.

A desvantagem desse esquema, no entanto, é que o usuário tem sempre que calcular assinaturas digitais para as compras, o que torna o processo muito lento. Além disso, a interferência de um terceiro elemento durante o processo de transação também é bastante indesejável.

SVP

O SVP [23], Small Value Payments, é um esquema totalmente baseado em funções de *hash*, em que a verificação das fichas pelo agente é feita *off-line*. O processo de pagamento é feito através da troca de desafios (números aleatórios) entre o vendedor e o usuário. O usuário aplica então a função de *hash* sobre os desafios em conjunto com uma ficha que foi inicialmente recebido do agente, criando um resultado cuja autenticidade o vendedor é capaz de verificar.

Os autores definem diversas políticas de segurança configuráveis, mas todas elas baseiam-se na presença de *hardware* dedicado e à prova de fraudes nas instalações dos vendedores. Esse requisito adiciona um custo muito grande ao projeto e está fora do escopo deste trabalho.

3.2 Propostas de melhoramentos

Alguns esquemas de micropagamentos foram propostos a partir de 1997 como melhoramentos sobre as propostas originais apresentadas acima. Identificaremos abaixo alguns deles, descrevendo-os brevemente.

NetCents

O protocolo NetCents [15] é um melhoramento proposto sobre o MilliCent. A idéia principal dessa proposta é fazer com que os títulos possam ser trocados diretamente entre vendedores, sem a presença do agente. Dessa forma o usuário não tem que revalidar seu título a cada troca de vendedor. Para que isso seja possível, o título foi dividido em duas partes, uma pública e uma privada. A parte pública contém uma chave pública utilizada pelos vendedores para verificação dos pagamentos e um balanço atual da conta do usuário; a parte privada contém uma chave privada utilizada para autenticar os pagamentos.

Embora reduza o número de intervenções do agente, o protocolo ainda possui a deficiência de fazer muito uso de algoritmos de criptografia assimétrica, como o RSA [19].

SPayWord, PPayWord e UPayWord

Em [11] os autores apresentam três protocolos para melhorar o esquema PayWord: o SPayWord, o UPayWord e o PPayWord.

O protocolo SPayWord foi desenhado para melhorar a segurança do PayWord e permitir que as fichas tenham tamanho menor. É baseado no princípio de que uma palavra pode ser acrescida à ficha antes de submetê-la à função de *hash*, alterando assim o resultado sem mudar a quantidade de computação (para uma determinada faixa de tamanhos de entradas, as funções de *hash* realizam a a mesma quantidade de computação). Essa palavra a ser acrescentada é o sal (de onde vem o S de SPayWord), e pode ou não ser conhecida de antemão pelo vendedor. Em cada caso o protocolo funciona de maneira ligeiramente diferente.

O protocolo UPayWord funciona sobre o PPayWord, e permite que o usuário gere

um número ilimitado de fichas a cada dia, o que é um melhoramento sobre o PayWord, em que o número máximo de fichas por dia tinha que ser fixado de antemão.

O protocolo PPayWord, em que o P vem de paralelo e que funciona sobre o SPayWord e o UPayWord, permite que uma ficha seja perdida sem que a próxima ficha deixe de ser validada. Segundo esse protocolo, a validação das fichas se dá diretamente com um valor derivado da ficha raiz.

Probabilistic Polling

No artigo [8] os autores introduzem a idéia de um sistema de micropagamentos que se situa no meio do caminho entre os sistemas *on-line* e os sistemas *off-line*. Segundo os autores os sistemas *on-line* introduzem caros custos e atrasos relativos à comunicação e são, por esse motivo, inadequados a transações de valores muito baixos. Já os sistemas *off-line* apresentam falhas de segurança que permitem a ocorrência de fraudes como a de gasto de fichas pelas quais não se pagou.

A proposta apresentada no artigo é a de envio probabilístico das fichas gastas em um vendedor para o agente. A probabilidade do envio de uma transação aumenta com o seu valor e também com o número de transações realizadas por usuário, segundo uma métrica definida pelo agente de acordo com a relação custo *versus* segurança desejada. Essa abordagem diminuiria a probabilidade de ocorrência de fraudes mantendo os custos de comunicação relativamente baixos.

MR1, MR2 e MR3

O artigo [10] apresenta três novos esquemas de micropagamentos: MR1, MR2 e MR3. Os três esquemas são melhoramentos sobre o esquema de loterias de Ronald Rivest [17].

A principal inovação presente nesse trabalho é o fato de que os autores fazem uso livre de assinaturas digitais sob a justificativa de que a evolução tecnológica possibilita essa computação com alto desempenho. Esse fato também foi utilizado neste trabalho, embora de forma muito moderada, como se verá adiante.

3.3 Implementações

Muitos trabalhos sobre micropagamentos apresentam implementações experimentais ou comerciais dos sistemas que propõem. Entre eles estão os sistemas MilliCent, NetCard e NetCents. Outros trabalhos sobre micropagamentos envolvem apenas a implementação de esquemas já propostos; é este o caso do Mimi [6], uma implementação Java do esquema MicroMint, e da dissertação [13], em que o autor apresenta uma implementação do protocolo PayWord.

Vistas algumas das principais propostas da área, passaremos no próximo capítulo a descrever a proposta apresentada neste trabalho, o sistema $\mu\mathbf{P}$.

Sistema	Característica	Vantagens	Desvantagens
Millicent [9]	- títulos específicos por vendedor;	- título mantém dados da conta do usuário;	- um título só vale para um vendedor;
NetCard [3]	- geração de moedas de 64 *bits*; - 4 protocolos;	1: geração no banco; 2: geração no usuário; 3 e 4: segurança;	1 e 2: uso de assinaturas digitais; 3 e 4: *hardware*;
PayWord [18]	- usuário recebe certificado e gera fichas;	- autenticação rápida;	- muitos conjuntos de fichas; - assinaturas digitais;
MicroMint [18]	- geração de colisões de *hash*;	- moeda facilmente autenticada;	- investimento grande em *hardware*;
Tick Payments [14]	- geração de fichas no cliente;	- abrangente	- proposta teórica; - *hardware* dedicado;
iKP [7]	- uso do protocolo iKP (IBM);	- relacionamento freqüente;	- intermediário; - assinaturas digitais;
SVP [23]	- verificação *off-line*;	- maior segurança;	- *hardware* dedicado;
NetCents [15]	- títulos trocados entre os vendedores;	- título único	- assinaturas digitais;
S,P e UPayWord [11]	- acréscimo de número aleatório ao *hash* (sal);	- segurança aumentada;	- as mesmas do PayWord;
probabilistic polling [8]	- envio probabilístico de fichas gastas ao agente;	- aumenta proteção contra fraudes;	- aumenta os custos de comunicação;
MR1, MR2 e MR3 [10]	- uso livre de assinaturas digitais;	- utilização de tecnologia recente	- esquema obscuro; - difícil aceitação;

Tabela 3.1: Comparação resumida dos trabalhos citados.

Capítulo 4

Arquitetura

4.1 Opções

O desenvolvimento da arquitetura do μ**P** foi um processo iterativo de muitas fases. Muitos problemas surgiram no decorrer da solução de outros, e as soluções, por sua vez, normalmente levavam a outros problemas. Muitos problemas de engenharia apresentam essa característica; o desejável, porém, é que o processo seja convergente, ou seja, que lentamente o número de problemas seja menor que o número de soluções. Esse foi, felizmente, o caso do μ**P** . Este capítulo descreve a arquitetura do sistema, mostrando como chegamos a cada uma das soluções. O processo de tomada de decisões pode ser acompanhado através da figura 4.1, na página 38.

A primeira decisão a ser tomada foi qual esquema subjacente seria utilizado na implementação do sistema. Alguns esquemas, como impunham a necessidade de utilização de *hardware* dedicado, foram descartados imediatamente (ver capítulo anterior); o uso de *hardware* dedicado, embora geralmente apresente melhor desempenho, encarece muito o projeto e é inviável para uma implementação experimental como a que estávamos nos propondo a fazer. A saída foi, então, buscar um esquema que fosse totalmente implementado em *software*.

Existe ainda outro motivo para implementar o sistema totalmente em *software*: o desenvolvimento da tecnologia de semicondutores, com a conseqüente queda nos preços dos *chips* de uso geral, pode levar sistemas de software mais rapidamente a patamares elevados de eficiência do que é possível fazer com sistemas de *hardware* específico e dedicado. Embora estes sejam intrinsecamente mais rápidos, a opção por *hardware* não-dedicado (microprocessadores como Athlon e Pentium) pode se beneficiar de seu rápido desenvolvimento e de sua grande flexibilidade.

Entre os esquemas totalmente baseados em *software* estudados, o mais simples é o PayWord [18]. Esse esquema é o mais claramente descrito por seus autores e o de

mais simples implementação, pois consiste de rotinas curtas e diretas. Além disso o PayWord é, ao lado do MilliCent, um dos esquemas mais estudados na literatura. Por esses motivos decidimos iniciar a implementação pelo esquema PayWord.

Uma das opções cruciais no início de qualquer implementação é a da tecnologia a ser utilizada. Existem inúmeras formas e linguagens de programação com as quais se pode construir um sistema, e essa escolha pode afetar fatores como o tempo de desenvolvimento, o número de pessoas envolvidas no projeto e também o resultado final, uma vez que cada tecnologia tem as suas limitações. No caso do μ**P** essa escolha teve como ponto central a questão da distribuição: o sistema deveria ser construído para funcionar livremente através da Internet, sobre as múltiplas plataformas que nela se encontram, especialmente os sistemas operacionais Microsoft Windows, Linux e MacOS, que são os principais sistemas em número de usuários no mundo. Por esse motivo foi escolhida a tecnologia Java, que oferece as seguintes vantagens:

- portabilidade entre as plataformas para as quais exista uma máquina virtual (existem máquinas virtuais para Microsoft Windows, Linux e MacOS);
- possibilidade de desenvolvimento uniforme de módulos cliente, servidor e programas *standalone* utilizando a mesma linguagem de programação e até as mesmas classes, o que reduz muito o tempo de desenvolvimento;
- familiaridade dos desenvolvedores com a linguagem de programação;
- disponibilidade de vasta documentação, exemplos e listas de discussão na Internet.

Escolhida a tecnologia, passou-se para a etapa de adaptar o esquema PayWord para a mesma. Nesse ponto surgiu a primeira e fundamental diferença entre o μ**P** e o esquema de Rivest: o Payword exige a computação de uma assinatura digital no usuário para cada talão de fichas emitido, e isso significa uma assinatura digital para cada vendedor para cada dia, pois um talão só tem validade por um dia em um vendedor. O cálculo de uma assinatura digital nessa quantidade pode não parecer muito se assumirmos que um usuário não deve visitar muitos vendedores por dia (essa é a hipótese dos autores do PayWord), mas nossa decisão foi de eliminar a necessidade dessa assinatura por dois motivos:

1. mesmo sendo ocasional, o cálculo de uma assinatura digital no usuário pode ser um grande problema, pois no ambiente distribuído da Internet não se pode contar com capacidade de processamento nas máquinas clientes; a diversidade encontrada nesses equipamentos é muito grande, e um bom projeto deve contemplar ao máximo essa diversidade;

2. a tecnologia Java apresenta séria restrição para o cálculo de assinaturas digitais nos clientes. Isso se deve ao fato de a versão da máquina virtual distribuída com o navegador mais popular do mercado, o Microsoft Internet Explorer, ser de uma versão que ainda não tinha as classes necessárias para o cálculo de assinaturas digitais. A versão mais atual da máquina virtual tem essas classes, mas seu *download* pode levar um tempo proibitivo em lugares com conexão lenta. O fato de ser distribuída uma máquina virtual antiga com um navegador moderno se deve exclusivamente a disputas de mercado entre a Microsoft e a Sun Microsystems. Assim, para que o sistema abarcasse o maior número de clientes possíveis, as *applets* desenvolvidas não poderiam calcular assinaturas digitais.

O próximo aspecto a ser discutido na arquitetura do μ**P** foi um ponto levantado como fraco no esquema PayWord: a necessidade de criação de um conjunto independente de fichas para cada vendedor. É certo que essa forma de funcionamento facilita bastante a implementação do sistema, como veremos a seguir, mas ela também apresenta uma desvantagem: implica no cálculo de um novo conjunto de fichas para cada vendedor visitado a cada dia, o que pode ser muito custoso computacionalmente para ser executado em *applets* nos computadores dos usuários.

Assim, a primeira idéia que surgiu para o μ**P** foi a idéia de fazer o que se chamou um sistema de micropagamentos específicos: o sistema seria desenhado para funcionar diretamente ligado a um vendedor; o conjunto de fichas geradas seria único e somente poderia ser gasto com aquele vendedor. Esse seria um sistema de implementação ainda mais fácil, e poderia ser utilizado como forma de pagamento em sistemas como o SCD-SID ou outros sistemas que necessitassem de uma forma específica de controle de pagamentos. Seu funcionamento se basearia no conceito de um cartão telefônico pré-pago: compra-se da empresa um número de créditos que se utilizará então somente nos apa-

relhos daquela empresa. Essa idéia funciona bem se a empresa que está vendendo os créditos tem força suficiente de mercado para que se saiba que todos ou quase todos os créditos comprados serão gastos mais cedo ou mais tarde. No entanto ao pensarmos em uma pequena empresa ou mesmo pessoa física vendendo conteúdo digital na Internet, a idéia de compra de créditos específicos cai por terra: dificilmente um consumidor compraria créditos que só poderia utilizar em um *site* da Internet se não tivesse a certeza de visitar muito freqüentemente esse *site*.

A partir desse problema surgiu a idéia de fazer o que se chamou de um sistema de micropagamentos genéricos: um sistema em que apenas um conjunto de fichas fosse criado, que pudesse ser utilizado em qualquer um dos vendedores cadastrados. Um esquema desse tipo se assemelha muito à emissão de uma moeda por parte do Estado: a força do emissor determina a aceitação ou não aceitação da moeda; a força da moeda depende da situação econômica do emissor; a utilização da moeda pelo público depende dos dois fatores anteriores e de um mais sutil, a confiança. Fica evidente que um sistema de micropagamentos genéricos é mais ousado e complexo, mas também que era a única alternativa que o projeto μ**P** poderia tomar para manter-se coerente com suas propostas. O sistema de micropagamentos específicos, no entanto, ainda poderia ser útil no caso de vendedores com grande força de mercado que quisessem utilizar o μ**P** em suas operações; ficou decidido, assim, por criar um sistema híbrido que possibilitasse as duas configurações, de acordo com o desejo do vendedor ao se cadastrar.

O sistema de micropagamentos genéricos imaginado funcionaria da seguinte forma: um único conjunto de fichas seria gerado para cada usuário no momento de seu cadastramento no sistema. Quando visitasse vendedores cadastrados o usuário poderia enviar as fichas para realizar compras. As fichas enviadas teriam então que ser verificadas localmente em cada vendedor. Como as fichas são valores de saída de uma função de *hash*, elas têm que ser verificadas através da aplicação da função sobre elas e posterior comparação com valor subseqüente. O importante nesse ponto é notar que o valor subseqüente na cadeia gerada já tem que estar armazenado no vendedor para que ele possa realizar a comparação. Assim ficou criado o problema de como enviar aos vendedores a ficha raiz (que é na verdade a última ficha da cadeia de *hash*) através da qual os pagamentos seriam autenticados.

A primeira abordagem para a solução desse problema foi a da criação de um certificado do vendedor, que seria enviado a todos os vendedores no momento da criação do certificado do usuário. Assim, se o certificado do usuário contivesse uma seqüência de n fichas, o certificado do vendedor conteria a ficha $n+1$ e informações sobre o usuário a quem pertenceria aquela ficha. Ao realizar a primeira compra, o usuário enviaria ao vendedor a ficha n; sobre ela o vendedor aplicaria uma vez a função de *hash* e obteria então a ficha $n+1$, que já estaria armazenada localmente. Se as fichas fossem iguais a compra estaria autorizada, e a ficha n seria então armazenada no vendedor. Na próxima compra, o usuário enviaria a ficha $n-1$, cujo *hash* é a ficha n, que agora também está armazenada e que portanto autentica a ficha $n-1$ enviada, e assim por diante.

Esse esquema parece funcionar muito bem, mas o que acontece se a ficha $n-1$ for enviada a outro vendedor, no qual a ficha n não foi gasta e portanto não está armazenada? A solução para esse problema seria aplicar duas vezes a função de *hash* sobre a ficha $n-1$, chegando então à ficha $n+1$ que consta do certificado do vendedor e que está, portanto, armazenada localmente. Assim, a ficha agora tinha que estar acompanhada por um número, a que se chamou ordem da ficha, que representa o número de vezes que a função de *hash* tem que ser aplicada sobre aquela ficha para que se chegue à ficha raiz. Ao receber uma ficha bastaria ao vendedor aplicar k vezes a função de *hash* sobre ela, onde k é o número de ordem enviado junto com a ficha, e compará-la com a ficha raiz constante no certificado daquele usuário.

Embora essa questão parecesse resolvida, persistia ainda um problema: a transmissão dos certificados a todos os vendedores no momento da geração de um certificado de usuário. Os certificados de usuários são normalmente gerados a intervalos de tempo razoáveis, mas existem alguns fatores que podem levar à geração de um número considerável de certificados:

1. o número de usuários do sistema pode simplesmente crescer, e com isso o número de certificados iniciais gerados também cresceria;

2. os usuários podem perder seus certificados (apagando-os por engano, ou tendo que formatar seus discos, por exemplo), e isso não pode significar que perderam os créditos pelos quais já pagaram. Assim, é preciso haver uma possibilidade de re-

geração de certificados, o que pode aumentar ainda mais o número de certificados gerados;

3. se um usuário possui um certificado com p ficha ainda não gastas e pretende realizar uma compra de $p+q$ fichas, ele precisará adquirir ao menos mais q fichas. Isso significa a geração de um novo certificado, contendo agora ao menos $p+q$ fichas. Esse fato pode ocorrer com freqüência, levando a um crescimento ainda maior no número de certificados gerados.

Podemos perceber agora que o número de certificados gerados por dia pode ser muito grande. Se considerarmos que a aceitação do sistema crescerá, então devemos supor que o número de vendedores cadastrados também crescerá. O número de envios de certificados de vendedores será igual ao número de certificados de usuários gerados multiplicado pelo número de vendedores cadastrados. Esse custo de comunicação pode ser proibitivamente alto.

Uma alternativa para realizar essa comunicação seria o uso de *multicast*, ou seja, apenas um envio da mensagem que alcançaria todos os vendedores. Há, no entanto, duas razões para que não se possa utilizar essa tecnologia nesse caso: a comunicação tem que ser segura (o que não é possível em *multicast*) e a recepção do certificado tem que ser confirmada, pois um vendedor sem certificado não poderia autenticar a compra de um cliente válido, o que seria uma falha inadmissível do sistema.

A alternativa escolhida para a solução do problema de comunicação foi a seguinte: ao invés de enviar certificados contendo a ficha raiz aos vendedores no momento da geração dos certificados dos usuários, a ficha raiz seria enviada junto com a ficha de compra a cada transação. Ao receber uma ficha para compra, o vendedor recebe junto com ela a ficha raiz, aplica sobre a ficha de compra k vezes a função de *hash* e compara com a ficha raiz que acabou de receber. Para que isso funcione, no entanto, a ficha raiz tem que apresentar uma garantia: a assinatura digital do agente. Ao gerar o certificado do usuário, o agente gera um pequeno certificado de vendedor contendo a última ficha da seqüência, assina esse certificado e o insere como mais um campo no certificado do usuário. Quando vai fazer uma compra, o usuário envia a próxima ficha a ser gasta e esse certificado, que contém a ficha raiz. Ao receber a mensagem o vendedor verifica

a assinatura digital do certificado que contém a ficha raiz através da chave pública do agente e pode assim autenticar a compra.

O inconveniente da solução apresentada acima é o fato de que o vendedor tem que verificar uma assinatura digital a cada transação. Esse fator certamente aumenta o atraso na conclusão de uma compra, mas ainda não é tão grande quanto a criação de uma assinatura digital. Além disso, pode-se assumir, como em [10], que o desenvolvimento tecnológico de *hardware* e *software* fez com que essa operação seja hoje mais acessível e que o equipamento dos vendedores é, em média, bastante superior em capacidade ao dos usuários, podendo suportar essa operação com razoável eficiência.

4.1.1 Proteção contra fraudes

O próximo problema a ser definido na arquitetura do μ**P** foi o seu esquema de proteção contra fraudes. Do ponto de vista do agente, que é quem vai pagar aos vendedores pelas fichas gastas apresentadas por eles, existem dois tipos de fraudes que podem ocorrer: a fraude do vendedor e a fraude do usuário.

A fraude do vendedor consiste na possibilidade de o vendedor criar e apresentar ao agente uma ficha válida que não tenha sido gasta em seu *site*. Essa fraude se tornou possível a partir do ponto em que as fichas podem ser gastas seqüencialmente em vendedores diferentes. É computacionalmente inviável ao vendedor descobrir a ficha n a partir da ficha $n+1$, pois $n+1$ corresponde a $h(h^n(x))$, onde h é a função de *hash*, escolhida justamente por ser uma função de mão única. Analisemos, no entanto, a seguinte situação:

1. o usuário gasta a ficha n com o vendedor A
2. o usuário gasta a ficha $n-1$ com o vendedor B
3. sabendo o valor da ficha $n-1$, B pode facilmente descobrir o valor da ficha n e apresentar essa ficha como tendo sido gasta em seu *site*

Nessa situação, A e B apresentam ao agente a mesma ficha, pedindo em troca o seu valor em dinheiro. Se não houver nenhuma informação extra, o agente não terá

como saber quem é o fraudador, e será obrigado a pagar ambas as fichas, arcando com o prejuízo.

A fraude do usuário, por sua vez, se baseia no fato de que o certificado do usuário, que contém todas as suas fichas e a ficha raiz assinada pelo agente, está armazenado localmente em seu computador. Para maior segurança, decidiu-se por armazenar esse certificado em forma criptografada. A criptografia utilizada é simétrica e a chave é derivada da senha do usuário. Isso impede que um usuário tenha acesso ao certificado do outro, mas não impede que um usuário mal-intencionado tenha acesso ao próprio certificado (desde conheça a forma de funcionamento do sistema, o algoritmo de criptografia utilizado, a forma de gerar a chave e o funcionamento geral do sistema) e introduza modificações que lhe sejam favoráveis, como por exemplo voltar o ponteiro que indica a próxima ficha a ser gasta; dessa forma o usuário pode gastar a mesma ficha n em vários vendedores diferentes. Nesse caso os vendedores apresentarão fichas que, em sua visão, foram legitimamente gastas em seus *sites*, obrigando o agente a honrar os pagamentos.

Outro tipo de fraude que o usuário pode cometer se baseia na geração de certificados: o usuário deve ter o direito de gerar um novo certificado em caso de perda. O novo certificado invalida o anterior e repõe as fichas não gastas. O problema, no entanto, é que o gasto e a geração de fichas se dão um lugares diferentes, o primeiro nos vendedores e a segunda no agente. Assim, se o agente ainda não sabe que uma determinada ficha já foi gasta por um usuário que está pedindo um novo certificado, ele pode gerar o novo certificado contendo essa ficha, que será gasta de forma legítima.

A geração de fichas além das que já existem dentro de uma seqüência é computacionalmente inviável também para o usuário, pois implica em descobrir a inversa da função de *hash*, e a geração de uma nova cadeia de fichas esbarra na impossibilidade de criação de uma assinatura digital válida sobre a ficha raiz, que só pode ser feita com a chave privada do agente.

Para se proteger das fraudes possíveis, o μ**P** se baseia em duas premissas, da mesma maneira que [9]:

- os valores envolvidos são muito baixos e não compensam o trabalho e custo envolvidos em burlar o sistema;

- a repetição de pequenas fraudes pode ser facilmente detectada e o fraudador descoberto e punido.

Apesar de considerar as premissas acima, o desenvolvimento de uma arquitetura de micropagamentos tem que contar com um esforço para diminuição de fraudes já previstas; um sistema que prevê e aceita a ocorrência de fraudes, embora conceitualmente correto, pode ter dificuldade em angariar a confiança dos seus participantes principais, os usuários e os vendedores. Em termos de *marketing*, um sistema que admite fraudes pode facilmente se tornar um fracasso.

Para que o agente impeça alguns tipos de fraudes e detecte outros, é necessário que ele faça a verificação das fichas gastas nos vendedores. Essa verificação pode ser feita de quatro formas diferentes:

1. ***on-line*** - a cada transação, quando recebe a ficha do usuário, o vendedor pode consultar diretamente o agente a respeito da validade da ficha. Isso garante proteção total contra fraudes, uma vez que o agente sabe de todas as fichas gastas e pode negar a autorização para uma transação com ficha já gasta. Essa opção já havia sido descartada anteriormente, no entanto, por dois motivos: impõe um atraso muito grande a cada transação e introduz um ponto único de falha imediata para todo o sistema.

2. ***off-line* com pesquisa** - o agente consulta regularmente os vendedores acerca das fichas gastas em cada um deles, e repete essa consulta, restrita às fichas gastas por um usuário específico, quando esse usuário pede um novo certificado. Esse esquema não impede que as fraudes sejam feitas mas permite que o agente, através do cruzamento dos dados dos vendedores, detecte a sua ocorrência. Não seria possível, no entanto, diferenciar entre a geração de fichas pelos vendedores e o gasto duplo por parte dos usuários. O problema que esse esquema apresenta é a enorme carga computacional e de comunicação necessária para realizar a pesquisa, o que pode inserir um atraso muito grande na geração de certificados e também exigir que o agente possua um equipamento muito caro para fazer todo o processamento ao fim de cada dia.

3. ***quase on-line*** - a cada transação o vendedor envia para o agente informações sobre a compra realizada (usuário, ficha recebida, ordem da ficha, valor da compra e horário), porém *sem aguardar a autorização do mesmo.* Esse esquema proporciona o conhecimento imediato acerca das transações por parte do agente sem, contudo, implicar em atraso na autenticação da compra, ou seja, é imperceptível para o usuário. Sua desvantagem é que a quantidade de mensagens chegando no agente ao mesmo tempo pode ser muito grande nos horários de pico de compras; o tamanho das mensagens é, no entanto, muito pequeno e seu processamento é simples. Embora não impeça o duplo gasto de fichas por parte de usuários fraudulentos, esse esquema detecta esse tipo de fraude rapidamente, podendo então cancelar a validade do certificado do usuário, como veremos adiante, e possibilita maior facilidade na distinção entre esse tipo de fraude e a fraude do vendedor, com base no horário de recebimento das fichas.

4. **em lotes** - nesse esquema o vendedor envia as fichas gastas para o agente em lotes, em períodos fixos ou variáveis de tempo. A configuração do período de tempo entre as remessas permite grande flexibilidade ao sistema: se o tempo é curto o sistema funciona como na opção 3, com o agente recebendo informações a cada transação; se o tempo é mais longo os custos de comunicação e processamento são menores e, em contrapartida, a possibilidade de fraudes aumenta. A configuração do tempo entre os envios dos lotes pode ser feita de duas formas: manual ou dinâmica. Na configuração manual o intervalo de tempo é um parâmetro do sistema que pode ser editado manualmente em um arquivo de configuração, e na configuração dinâmica existe um algoritmo que determina esse intervalo de tempo de acordo com a movimentação atual de transações no vendedor, seguindo a idéia de [8].

A solução escolhida para o μ**P** foi a número 3, quase *on-line*, mas decidimos manter a possibilidade de realizar o envio das fichas em lotes, como na solução 4. A utilização de um esquema ou de outro depende apenas de um parâmetro configurável em tempo de execução nos vendedores. O agente, por sua vez, não vê diferença entre as duas

formas, apenas processa as fichas recebidas através de uma *servlet* sem saber se essas fichas foram gastas imediatamente ou algum tempo antes.

4.1.2 CRL

Até agora discutimos como o agente pode saber da ocorrência de fraudes e precaver-se contra elas. Há dois aspectos, entretanto, que necessitam maior esclarecimento: como os vendedores podem se precaver contra fraudes e como o agente pode punir os vendedores ou usuários fraudulentos.

Os vendedores recebem fichas que vêm de certificados dos usuários, acompanhadas de fichas raiz assinadas que as autenticam. Cada certificado contém apenas uma ficha raiz. Existem alguns casos nos quais um certificado deixa de ser válido:

- quando o usuário cometeu uma fraude;
- quando o usuário requisitou um novo certificado, seja por ter perdido o antigo, seja para comprar mais fichas.

A requisição de um novo certificado, mesmo que o antigo não esteja expirado e contenha fichas não gastas, deve revogar (ou invalidar) automaticamente todos os certificados anteriores, de forma que haja apenas um certificado de micropagamentos genéricos válido a qualquer instante.

Quando um certificado é revogado, no entanto, ele pode ainda estar armazenado no computador de um usuário. A informação sobre a sua revogação está somente no banco de dados do agente. Como a ficha raiz que acompanha o certificado também está armazenada no computador do usuário, este ainda é capaz de realizar uma compra, pois o vendedor ainda não possui a informação de que o certificado não está válido e a ficha raiz é autêntica quando assinada pelo agente.

Assim, é preciso encontrar uma forma de avisar aos vendedores quando um certificado é revogado. Em sistemas de segurança os certificados revogados são mantidos em uma lista chamada CRL (*Certificate Revokation List* ou lista de revogação de certificados). A CRL, no entanto, deve ser acessada por todos os vendedores no momento da verificação de uma ficha. Para atingir esse objetivo foram estudadas três propostas:

1. o vendedor verifica a CRL junto ao agente a cada transação; essa solução é inviável pois implica em um atraso muito grande sobre a compra, diminuindo muito a eficiência do sistema;

2. o agente avisa todos os vendedores sobre a entrada de um novo certificado na CRL; essa solução implica em um custo muito grande de comunicação, que aumenta com o aumento do número de certificados gerados;

3. o vendedor verifica a CRL junto ao agente de forma periódica; essa é uma solução de compromisso entre o custo de comunicação e a imunidade a fraudes: quanto maior for o intervalo, menor será o custo e maior a probabilidade de que um certificado que já tenha sido revogado seja utilizado para uma transação. Essa foi a solução escolhida para o sistema μ**P**

4.1.3 Integração

O próximo passo a decidir sobre a arquitetura do μ**P** foi a sua integração a outros sistemas; como fazer para que, ao final de um processo de compra, o produto seja realmente entregue ao usuário?

O processo de compra acaba quando o vendedor compara a ficha raiz com o resultado da n-ésima aplicação da função de *hash*. Esse resultado é binário: se forem iguais a compra é autorizada e, se não forem, é cancelada. Como quem controla a compra do lado do usuário é uma *applet*, ela pode receber de volta a resposta do vendedor e redirecionar o navegador do usuário para a página do produto, caso a transação tenha sido autorizada.

A página do produto, no entanto, não pode estar disponível abertamente, pois nesse caso bastaria a um usuário mal-intencionado descobrir esse endereço e utilizar a página sem passar pelo processo de pagamento. Assim, concluímos que o produto a ser vendido deve estar inserido em uma página dinâmica, que verifica se o usuário que está pedindo a página está ou não autorizado a obtê-la.

O problema que surge nesse ponto é como será implementada essa página dinâmica. Páginas desse tipo podem ser implementadas com as mais diversas tecnologias, e não

é a intenção de projeto de micropagamentos restringir a forma de montagem de uma página de produto somente pelo fato de que ele tem que ser pago através desse sistema.

Assim, optamos por utilizar uma solução de integração que pode ser adequada de forma simples a qualquer implementação de página dinâmica: o banco de dados. Ao finalizar o processo de compra e verificar a autenticidade da ficha, o vendedor simplesmente gera uma autorização aleatória e insere essa autorização no banco de dados, em uma tabela específica e com validade de alguns segundos ou minutos. Dessa forma, a *applet* tem apenas que receber a autorização e redirecionar o navegador para a página do produto acrescentando ao cabeçalho HTTP a autorização recebida. A página dinâmica que fornece o produto ao usuário tem somente que consultar o banco de dados e verificar se a autorização é válida.

4.2 Funcionamento

A partir das opções listadas na seção anterior, pode-se resumir o μ**P** através das seguintes características:

- é totalmente baseado em *software*;
- implementado com tecnologia Java em clientes e servidores;
- é capaz de realizar micropagamentos específicos ou micropagamentos genéricos, dependendo da vontade do vendedor sobre a forma sob a qual quer operar;
- realiza a autenticação da transação de forma *off-line*;
- tem proteção moderada contra fraudes, que funciona no esquema *quase on-line* ou em lotes;
- possibilita a integração com outros sistemas através de banco de dados.

4.2.1 Componentes

O sistema μ**P** é composto por cinco elementos:

1. a **ficha** é um valor de 20 *bytes* codificado em Base64 e que possui um número serial ou de ordem associado;

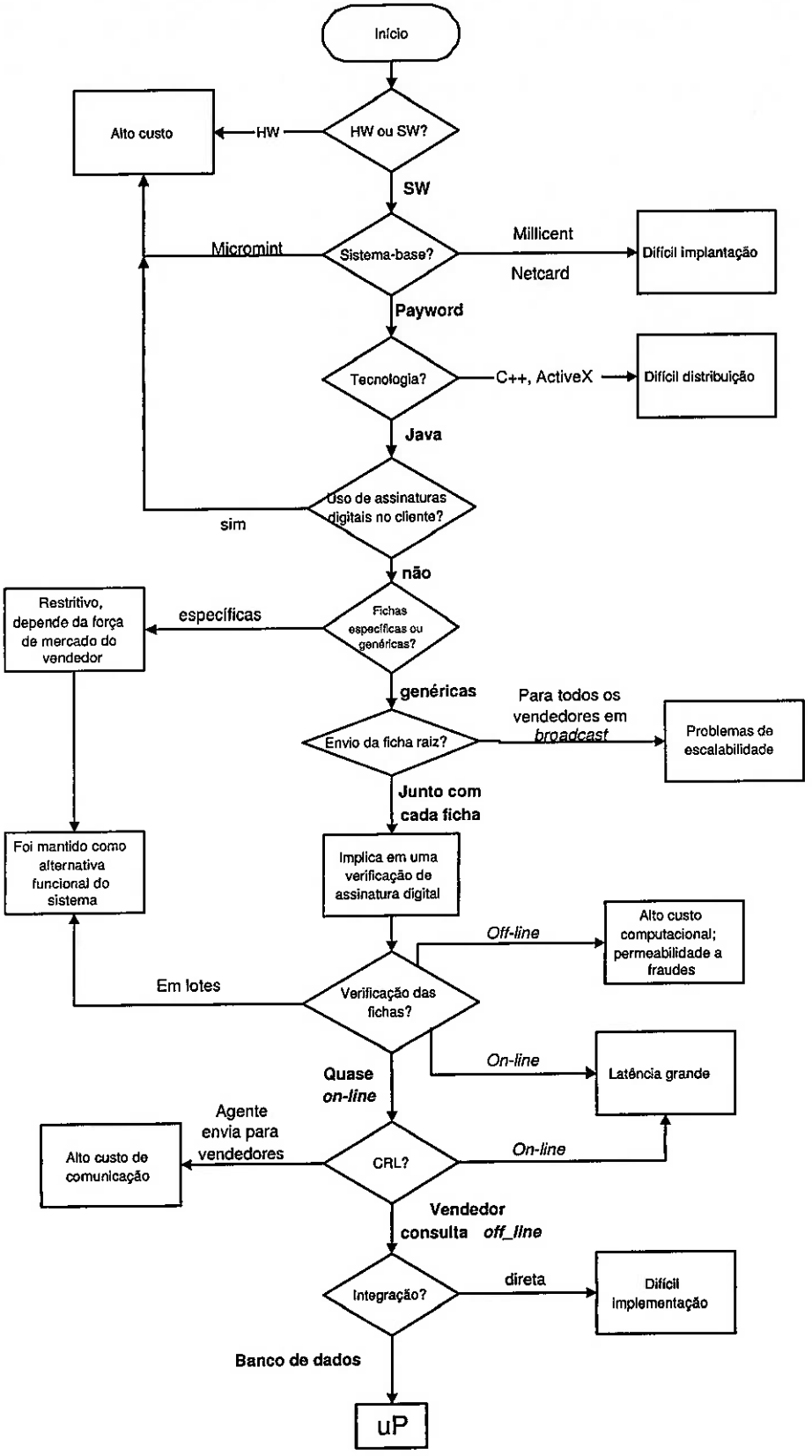


Figura 4.1: Fluxograma do processo de tomada de decisões.

2. o **certificado** é um conjunto de fichas seqüenciais (cada ficha é o resultado da aplicação da função de *hash* sobre a ficha anterior; a ficha inicial é aleatória) e ordenadas, e contém os seguintes elementos:

   - um número serial;
   - uma data de expiração;
   - o nome do usuário ao qual pertence;
   - o nome do vendedor ao qual se destina (a palavra-chave *genérico* indica um certificado que pode ser gasto com qualquer vendedor que se cadastre nessa categoria);
   - o nome do agente;
   - o número total de fichas do certificado;
   - o valor (financeiro) individual das fichas;
   - um ponteiro indicando qual a próxima ficha a ser gasta;
   - a própria seqüência de fichas;
   - um valor de verificação de integridade do conteúdo, que nada mais é que um *hash* simples do mesmo e
   - a ficha raiz da seqüência, assinada digitalmente pelo agente para garantia de sua autenticidade e codificada em Base64;

3. o módulo do **usuário**;

4. o módulo do **vendedor** e

5. o módulo do **agente.**

Certificado

O certificado é gerado pelo agente a pedido do usuário e é armazenado no banco de dados do primeiro e no disco do segundo. O certificado é armazenado no disco local do usuário em forma criptografada, com a chave de criptografia simétrica sendo gerada a partir da senha do usuário.

A cada momento existe apenas um certificado válido por usuário por vendedor. Como o sistema μ**P** permite a geração de certificados para micropagamentos genéricos ou específicos, pode haver certificados de micropagamentos específicos para cada vendedor cadastrado sob essa forma, além de um único certificado de micropagamentos genéricos.

Usuário

O usuário interage com o sistema μ**P** em três momentos: o de seu cadastramento, o da compra e o do pedido de novo certificado.

O cadastramento é feito através do acesso direto a uma página do agente que fornece esse serviço, através de uma conexão segura via SSL. No momento do cadastramento o usuário fornece a senha com a qual serão criptografados os seus certificados.

Quando vai fazer uma compra o usuário tem que baixar e executar uma *applet* que tem as seguintes funções:

1. procurar no disco local pelo certificado do usuário;
2. decriptografar o certificado e verificar a sua integridade;
3. verificar se o certificado possui o número de fichas não-gastas necessário para a compra;
4. em caso positivo, enviar a próxima ficha — de acordo com o valor da compra — para o vendedor, receber deste uma autorização e redirecionar o navegador do usuário para a URL do produto, anexando a autorização; em caso negativo, redirecionar o navegador para uma página de pedido de novas fichas no *site* do agente;
5. enviar, junto com cada ficha enviada ao vendedor, a ficha raiz, para que o vendedor autentique a ficha junto a esta.

Se o usuário não está cadastrado no sistema μ**P** , não possui localmente o certificado ou não possui o número de fichas necessário para a compra, a AppletUsuario redireciona seu navegador automaticamente para o agente, onde ele tem a opção de comprar um novo certificado ou gerar novamente o certificado que já possui.

Vendedor

O vendedor interage com o sistema em três momentos: o de seu cadastramento, o da venda de um produto, quando recebe e autentica uma ficha, e o do envio de informações sobre movimentação para o agente.

O cadastro do vendedor é o momento em que ele fornece seus dados para o agente e define um parâmetro crucial: se quer utilizar o sistema de micropagamentos específicos ou o sistema de micropagamentos genéricos. Essa interação não foi implementada; por enquanto, é feita manualmente através da manipulação de registros no banco de dados do agente.

No momento da transação, quando recebe uma ficha — que vem sempre acompanhada da ficha raiz assinada pelo agente, o vendedor realiza as seguintes tarefas:

- verifica a assinatura digital da ficha raiz;
- se a ficha não é autêntica, cancela imediatamente a transação, enviando de volta para a *applet* o código -1;
- se a ficha raiz é autêntica, calcula k vezes a função de *hash* sobre a ficha enviada, onde k é o número de ordem da ficha, e compara o resultado com a ficha raiz. Se forem iguais a transação é aprovada e o número de autorização é gerado; em caso contrário a transação é cancelada;
- em caso de transação realizada, envia para o agente a ficha gasta e outras informações pertinentes. Esse envio é configurável, podendo ser imediato (maior custo e maior resistência a fraudes) ou em lotes (menor custo e menor resistência a fraudes);
- insere a transação realizada em sua tabela local de histórico.

O envio de informações de movimentação para o agente pode acontecer de forma *quase on-line* ou em lotes. No primeiro caso a informação é enviada para o agente imediatamente após a autorização de uma transação; no segundo caso, é enviada em intervalos de tempo determinados por um algoritmo no programa de envio, que depende basicamente da quantidade de movimentação financeira realizada no último intervalo de tempo.

Agente

O agente atua no sistema como elemento central: é ele o emissor das fichas, quem coleta o dinheiro dos usuários em troca delas no início do processo e quem paga os vendedores em troca das fichas gastas ao final do processo. Ao agente cabe manter um registro de gastos atuais de cada usuário, além de uma cópia dos certificados e fichas emitidos, para que possa verificar a ocorrência de fraudes e tomar as medidas cabíveis: revogação de certificados e suspensão de usuários ou vendedores fraudulentos.

O agente recebe a informação de cada ficha gasta de duas maneiras diferentes, que podem ser escolhidas na configuração do sistema.

A primeira maneira é o envio imediato, por parte do vendedor, de cada ficha gasta; essa maneira tem a vantagem de manter o agente sempre atualizado com relação às fichas gastas no sistema, diminuindo muito a possibilidade de fraudes, sob o custo de aumentar a carga de comunicação e de computação do servidor do agente. É importante ressaltar, no entanto, que o envio da ficha se dá independentemente da resposta da transação ao usuário. Essa funcionalidade foi implementada através de uma nova *thread* (linha de execução), o que possibilita que a resposta da transação para o usuário não seja atrasada por esse processo.

A segunda maneira é o envio das fichas gastas em lotes; de tempos em tempos (esse intervalo também é configurável) o vendedor envia um lote com as últimas fichas gastas. Esse esquema concentra o envio de fichas, diminuindo os custos de comunicação e computação do servidor do agente, mas tem a desvantagem de que o agente fica desatualizado sobre gastos de fichas enquanto esses gastos ainda não foram enviados.

A atualização do agente sobre os gastos é importante por dois motivos: para a geração de novos certificados e para a detecção de fraudes.

A geração de novos certificados pode se dar quando o usuário necessita de mais fichas do que possui para realizar uma compra ou quando o usuário perdeu o certificado (isso pode acontecer quando ele muda de computador ou formata seu disco, por exemplo). O agente leva sempre em conta as fichas já possuídas pelo usuário na hora de gerar um novo certificado, mesmo que ele tenha pago por mais fichas. Assim, se o usuário possui p fichas e comprou mais q fichas, seu novo certificado terá $p + q$ fichas. Como

essa contagem considera apenas as fichas ainda não gastas, é importante que o agente saiba o mais cedo possível que uma ficha foi gasta.

A detecção de fraudes acontece quando uma ficha é gasta mais de uma vez, ou quando uma ficha não existente na base de dados é gasta. Em ambos os casos, o certificado ao qual aquela ficha pertence é colocado em uma lista de rejeição (CRL - *Certificate Revokation List*), que é consultada de tempos em tempos pelos vendedores. Também por esse motivo é importante que o agente saiba quando uma ficha foi gasta.

4.2.2 Processos

O sistema μ**P** apresenta os seguintes processos de funcionamento:

- cadastramento
- geração de fichas
- transação de compra de produto
- informe e verificação de transações pelo agente

cadastramento

O sistema μ**P** não foi projetado originalmente para prover anonimidade aos usuários. Cada usuário que utiliza o sistema deve fornecer inicialmente seus dados, incluindo dados únicos como o CPF, que pode ser verificado junto à Receita Federal para impedir duplicações e falsidade ideológica.

Além de dados pessoais como nome, endereço e telefone, o usuário tem que fornecer também um nome de usuário (também conhecido como *login* ou *username* - uma palavra que o identifica unicamente no sistema) e uma senha, que será utilizada em todas as operações de compra ou geração de certificados, para garantir a sua identidade, e também na criptografia de seu certificado local, para garantir sua privacidade. Por esse motivo, o processo de cadastramento deve ser feito sobre uma conexão segura SSL.

O cadastramento deve também ser realizado pelos vendedores. Embora não esteja implementado, no cadastramento os vendedores devem informar ao agente, além de outros dados, se querem utilizar o sistema de micropagamentos específicos ou genéricos.

geração de fichas

O processo de geração de fichas acontece quando o usuário precisa de fichas para realizar uma compra e não as tem em número suficiente; isso pode se dar caso o usuário possua um certificado com número insuficiente de fichas, caso tenha perdido seu certificado ou caso ainda não possua um certificado.

O processo pode ser acompanhado através da figura 4.2, à página 45, nas letras A a D:

A - pedido de fichas Durante essa fase o usuário informa ao agente o seu nome de usuário, sua senha e o número de fichas que quer comprar.

B - seqüência de pagamento Essa fase envolve protocolos de pagamento externos ao sistema em questão; é a fase em que o usuário utiliza seu cartão de crédito, por exemplo, para pagar por todas as fichas que está adquirindo. Como essa fase não faz parte do escopo do projeto, ela não será detalhada. Para maiores informações podem ser consultadas as referências [22] e [2].

C - geração de fichas A geração de fichas é feita pelo agente através dos seguintes passos:

1. confere a senha fornecida pelo usuário com a que está armazenada no banco de dados;
2. consulta no banco de dados o número de ordem k da próxima ficha não gasta;
3. gera um número aleatório e, a partir dele, a seqüência de fichas, com ordem começando em k;
4. assina digitalmente a última ficha da sequência, gerando um certificado de vendedor, e insere esse certificado ao final do certificado do usuário;
5. revoga o último certificado válido do usuário;
6. insere o certificado gerado no banco de dados;
7. disponibiliza o certificado para ser retirado pelo usuário em um diretório temporário.

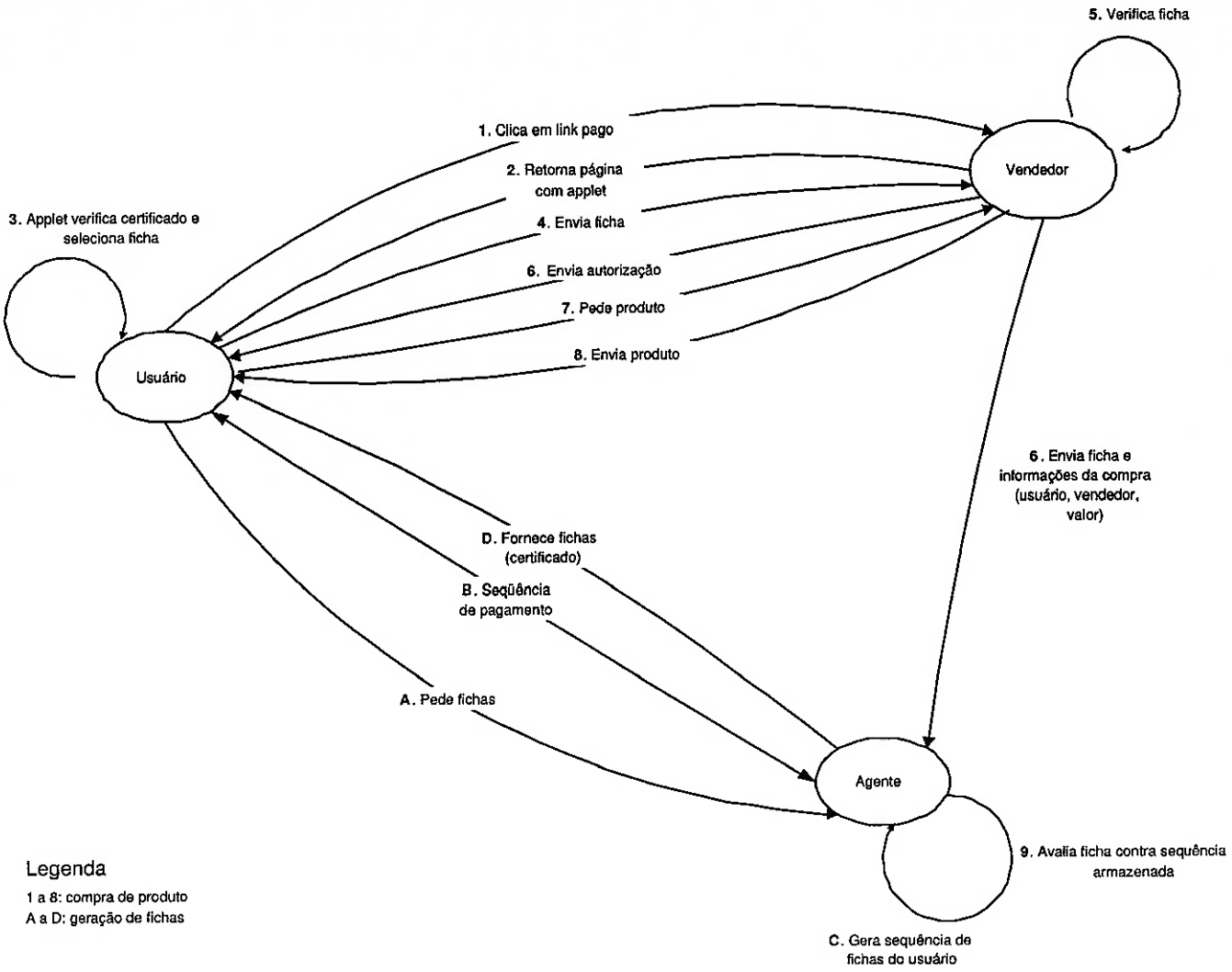


Figura 4.2: Fluxo do esquema μ**P**

D - entrega A entrega do certificado é feita através da carga de uma *applet* que recupera o certificado disponibilizado na fase anterior e o grava no disco local do computador do usuário, em um local pré-determinado.

compra de produto

A compra de um produto é a principal etapa de funcionamento do sistema, e pode ser acompanhado pelos passos 1 a 8 da figura 4.2:

1. A transação começa quando o usuário demonstra interesse em adquirir um produto, clicando em um *link* na página do vendedor. Supõe-se que o usuário chegou a essa página através de outros meios.

2. O sistema do vendedor retorna ao usuário uma página que contém a *applet* do sistema μ**P** . Essa *applet* é um o componente principal do módulo do usuário.

3. A *applet* busca o certificado local segundo as regras de nomenclatura (ver próximo capítulo), pede a senha ao usuário, decriptografa o arquivo e verifica se o certificado está válido e se possui o número necessário de fichas para fazer a compra. Em caso negativo, redireciona o navegador do usuário para uma página em que é oferecida a opção de compra de fichas ao usuário, passando assim ao passo A.

4. Caso o certificado seja válido, envia ao vendedor a ficha, sua ordem k, a ficha raiz assinada e outras informações pertinentes à compra: número serial do certificado, nome de usuário, nome do produto e valor da compra.

5. O vendedor verifica se o certificado não foi revogado, verifica a autenticidade da ficha raiz, calcula k vezes a função de *hash* sobre a ficha e compara o resultado com a ficha raiz.

6. Se os valores forem iguais, gera um número aleatório de autorização, insere esse número no banco de dados e o transmite de volta ao usuário; se forem diferentes, retorna ao usuário uma autorização de valor -1, indicando que a transação foi cancelada.

   Logo após o envio da autorização para o usuário o vendedor transmite informações sobre a transação (nome do usuário, nome do vendedor, ficha gasta, ordem da ficha, valor da compra e número da autorização) para o agente.

7. De posse da autorização, a *applet* redireciona o navegador do usuário para a página do produto, submetendo conjuntamente o número da autorização.

8. A página dinâmica que fornece o produto ou serviço adquirido verifica a autorização recebida junto ao banco de dados; se ela corresponder ao produto correto e estiver ainda válida, o produto ou serviço é então enviado ao usuário.

informe e verificação de transações

As informações sobre cada transação podem ser enviadas imediatamente após a verificação das mesmas, como descrito acima, ou em lotes a intervalos de tempo definidos.

A utilização de uma opção ou da outra depende de parâmetros configuráveis no vendedor.

Um programa *standalone* é responsável por varrer a tabela de histórico do vendedor e verificar se há fichas não enviadas; em caso positivo ele as envia em um só lote. O tempo que esse programa levará até a próxima varredura pode ser fixo ou variável. No caso de intervalo variável, o tempo entre as varreduras pode ser definido através da quantidade de movimentação ocorrida no último período. Embora esse esquema atrase a varredura sempre em um período em relação à movimentação, ele pode ajudar a manter o agente mais atualizado em períodos prolongados de grande movimentação.

No agente o programa que recebe esses dados é uma *servlet* que, como se verá no próximo capítulo, pode receber as informações de fichas avulsas ou em lotes, permitindo que esse parâmetro seja configurável apenas nos vendedores.

4.3 Comparação

Após o processo de definição da arquitetura do μ**P** , observamos que o resultado final é consideravelmente diferente da arquitetura do PayWord, em que foi baseada. As diferenças entre os dois sistemas são mostradas resumidamente na tabela 4.1.

PayWord	μP
um certificado para cada vendedor para cada dia	certificado único para todos os vendedores com validade configurável; aceita também certificados específicos por vendedor, porém com validade estendida
uso de assinaturas digitais no computador do usuário	usuário não calcula nenhuma assinatura digital
verificação da ficha através de aplicações da função de *hash*	verificação da ficha através de aplicações da função de *hash* e de uma verificação de assinatura digital
fichas geradas pelo usuário	fichas geradas pelo agente
vendedor informa ao agente somente a última ficha gasta de cada seqüência	vendedor informa ao agente sobre todas as fichas gastas em seu *site*
adequado para relacionamento freqüente entre usuário e vendedor	adequado para relacionamento ocasional entre usuário e vendedor
agente compara fichas com certificados gerados pelo próprio usuário	agente compara fichas com base de dados central
baseado em crédito	baseado em debito (pré-pago)

Tabela 4.1: Comparação entre o μ**P** e o PayWord

Capítulo 5

Implementação

5.1 Tecnologia

O sistema μ**P** foi desenvolvido utilizando duas tecnologias principais: Java e XML.

A tecnologia Java foi escolhida por ser multiplataforma — um requisito importante quando vários programas serão executados nos equipamentos dos vendedores e dos usuários, que podem ser dos mais variados tipos e portes — e por permitir que os programas de todos os módulos, tanto de clientes como de servidores, fossem implementados na mesma linguagem, possibilitando o reaproveitamento de código e agilizando o trabalho.

A tecnologia XML foi utilizada em todas as mensagens do sistema. Suas vantagens são: o fato de as mensagens serem facilmente lidas por seres humanos, o que facilita sua montagem e interpretação manuais, além da detecção e correção de erros; o fato de existirem interpretadores integrados com a linguagem Java, que precisam apenas ser configurados para as mensagens em questão e o fato de ser uma tecnologia em franca expansão, com muita documentação e discussões disponíveis na Internet.

5.2 Módulos

Os módulos que compõem o sistema são: agente, vendedor e usuário. Cada um desses módulos é composto por vários programas e por uma base de dados, exceto pelo módulo do usuário, que não apresenta base de dados mas apenas um certificado armazenado no computador local. A interação entre os diversos programas do sistema μ**P** descritos a seguir pode ser melhor acompanhada através da figura 5.2, na página 64.

5.2.1 Agente

O módulo do agente é composto por seis *servlets*[1] e uma base de dados.

Programas

As *servlets* que compõem o módulo do agente são:

ServAgCadastro Essa *servlet* é responsável por receber os dados de um novo usuário e inseri-los na base de dados. Cada usuário é único no sistema, e para garantir essa unicidade a *servlet* realiza a verificação de dois campos: o nome de usuário (diferente do nome da pessoa) e o CPF. Se o CPF já existe o usuário é impossibilitado de se cadastrar. Se o nome de usuário já existe ele é requisitado a fornecer outro, como acontece na maioria dos sistemas de computação.

A *servlet* de cadastro pede também a senha que o usuário utilizará sempre que tiver contato com o sistema μ**P**. Por esse motivo é necessário que a comunicação com essa *servlet* se faça sobre uma conexão SSL, embora essa funcionalidade ainda não tenha sido implementada.

ServAgPedeFichas Essa *servlet* tem a função de oferecer ao usuário algumas opções para compra de fichas. Ela é chamada quando o usuário tentou realizar uma compra e não conseguiu por não ter fichas suficientes ou por não possuir um certificado na máquina que estiver utilizando.

A **ServAgPedeFichas** recebe como parâmetros o nome do usuário, o nome do vendedor e o número de fichas necessário para realizar a compra, e fornece como resposta uma página em que são listadas as opções de compra de fichas.

O número mínimo m de fichas que o usuário pode comprar é definido por um parâmetro configurável no vendedor. Se o número de fichas necessário para a compra for menor que m, a compra mínima oferecida será de m fichas. Isso se deve ao fato de não ser possível efetuar compras de valores muito baixos utilizando

[1] Uma *servlet* é um pequeno programa servidor que recebe um pedido HTTP com parâmetros, realiza algum processamento e envia como resposta uma página HTML. É uma forma de se construir páginas dinâmicas com a tecnologia Java.

sistemas de pagamento convencionais, que são usados para a compra das fichas. As outras opções de compra são sempre múltiplos de m.

Outra opção oferecida, quando aplicável, é a de geração de novo certificado sem pagamento. Essa opção pode ser utilizada pelo usuário caso tenha perdido um certificado válido (em caso de formatação de seu disco, por exemplo). Embora essa opção dê margem à ocorrência de fraudes, considerou-se que o armazenamento de certificados no disco local do usuário não é uma forma segura, e que não seria justo obrigá-lo a manter toda a contabilidade de sua conta em um único arquivo que poderia ser perdido com facilidade.

A **ServAgPedeFichas** chama a **ServAgFichas**, descrita a seguir, para a geração das fichas.

ServAgFichas Essa é a principal *servlet* do módulo do agente: é a responsável por gerar as fichas e o certificado do usuário. Ela recebe como parâmetros o nome do usuário, sua senha, o nome do vendedor e o número de fichas requerido. Seus passos para a geração do certificado são:

1. confere a senha do usuário;
2. verifica o último certificado válido do usuário; se ele existir, guarda a ordem k da próxima ficha a ser gasta. Essa será a ordem da primeira ficha do certificado a ser gerado; dessa forma, as fichas de um usuário seguem sempre uma ordenação crescente, mesmo com a geração de mais de um certificado sobre as mesmas fichas compradas;
3. invalida o último certificado do usuário, que era o único válido, para o vendedor em questão (lembrando que o nome "`generico`" está reservado para a utilização do sistema de micropagamentos genéricos). A cada momento só pode existir um certificado válido por usuário por vendedor;
4. cria na base de dados a entrada para um novo certificado com data de expiração marcada para dali a um ano e recebe como resposta o número serial único desse certificado;
5. de posse da data de expiração, do número serial e do número de ordem

da próxima ficha, cria o certificado através de uma chamada para a classe **Certificado**. Essa é uma classe auxiliar que cria o certificado a partir de uma semente aleatória e o retorna em formato de *String*;

6. insere o certificado e as fichas criadas na base de dados local;
7. disponibiliza o certificado para *download* do usuário. Esse passo existe para que seja possível transferir automaticamente um arquivo para o computador do usuário. A transferência é feita da seguinte forma: o certificado é disponibilizado como um arquivo temporário em um diretório do servidor Web do agente, com um nome composto por uma concatenação de um número aleatório com o nome do usuário, seguido pela terminação `.scdsid` (por exemplo, `87443pedro.scdsid`. Esse nome, que forma junto com o endereço do servidor Web uma URL, é passado como parâmetro para a *applet* que irá instalar o certificado (chamada de **AppletCert**). A *applet* acessa o arquivo, grava-o no disco local do usuário e acessa então outra *servlet*, a **ServAgApaga**, para apagar o arquivo temporário.

ServAgApaga Essa *servlet* é responsável por apagar um arquivo de certificado do diretório de arquivos temporários onde ele foi disponibilizado, como explicado acima. Ela recebe como parâmetro o nome do arquivo a ser apagado e lê de um arquivo de configurações o nome do diretório em que ele se encontra. Sua resposta é vazia, e não é esperada pela *applet* que a chamou, pois sua ação é unilateral: se o arquivo não for apagado não há nada que a *applet* possa fazer. Nesse caso basta uma inspeção humana do diretório para verificar se há arquivos antigos que não foram apagados e a remoção pode ser feita manualmente.

ServAgHistorico Essa *servlet* tem a função de receber do vendedor a informação de que uma ficha foi gasta por determinado usuário e de inserir essa informação na base de dados do agente. Ela verifica também se existe alguma fraude relacionada à ficha recebida e toma as providências necessárias caso isso aconteça.

A forma básica de reconhecer uma fraude é a tentativa de uso de uma ficha já gasta ou não existente. Cada ficha recebida é conferida com a sua correspondente na base de dados, através dos números de certificado e de ordem. Caso haja uma

fraude, um contador de fraudes para o usuário é incrementado. Ao passar de um determinado limite (três fraudes, por exemplo), o valor do contador dispara um processo de cancelamento do certificado, executado por essa mesma *servlet*, que marca todas as fichas do usuário como gastas, o que impede a geração de um novo certificado sem o respectivo pagamento. Se o contador passar de outro limite pré-determinado (seis, por exemplo), o próprio usuário é marcado como inválido e impedido de gerar ou comprar novos certificados.

A **ServAgHistorico** foi projetada para funcionar com os dois modos de transmissão de fichas do vendedor: imediato ou em lotes. A mensagem que ela recebe está no formato XML, e um dos campos indica a quantidade de fichas recebidas. Dessa forma a configuração sobre o modo de operação pode ser feita apenas no vendedor.

Essa *servlet* emite também uma resposta no formato XML, que tem o intuito de informar ao vendedor sobre o recebimento e processamento de cada ficha. Assim, se houver algum problema com relação ao processamento de uma ficha, o vendedor a marca como não enviada, podendo enviá-la novamente mais tarde.

ServAgCRL Essa *servlet* é responsável por pesquisar a base de dados do agente e fornecer como resposta uma lista com todos os certificados revogados. Ela aceita parâmetros para fazer a pesquisa por data inicial e/ou final de geração, por usuário ou por vendedor, retornando uma lista de números seriais de certificados inválidos no formato XML.

Base de dados

A função da base de dados do agente é manter as informações sobre todas as fichas e certificados gerados para possibilitar a autenticação e posterior pagamento das transações, além da detecção e tratamento de possíveis fraudes. A base de dados é também o repositório central do sistema com relação a usuários e vendedores, permitindo a administração de suas contas.

Como a implementação do sistema μ**P** ainda é experimental, faltam campos e tabelas que seriam necessários a um sistema comercial (como exemplo podemos citar

informações relativas às contas financeiras de usuários e vendedores, que no momento não existem).

A base de dados do agente, como se vê na figura 5.1, é composta por seis tabelas:

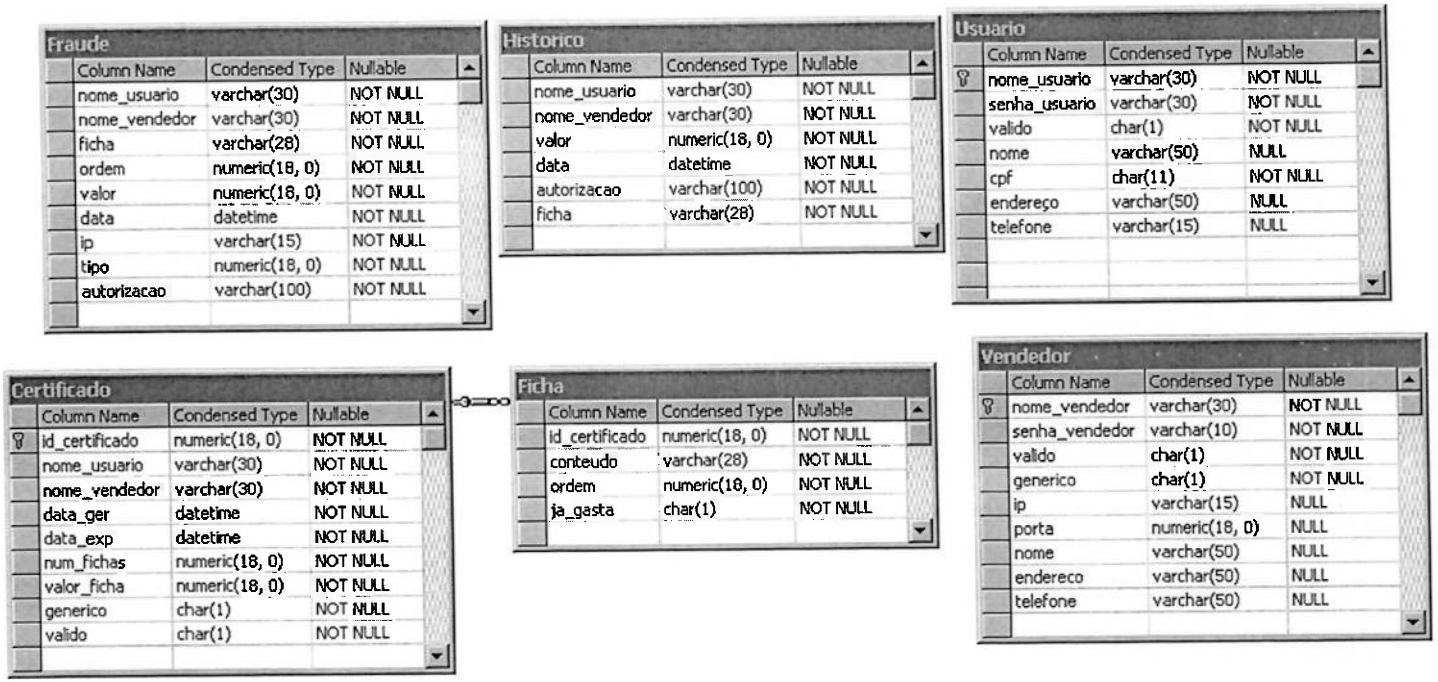
Usuario Contém todas as informações referentes ao usuário; seus campos são:

- `nome_usuario` - único no sistema;
- `senha_usuario` - armazenada de forma criptografada;
- `valido` - indica se o usuário é ou não válido;
- `nome` - o nome real do usuário;
- `cpf` - o CPF do usuário, único no sistema;
- `endereco` - o endereço do usuário;
- `telefone` - o seu telefone.

Vendedor Contém as informações referentes ao vendedor. Seus campos são:

- `nome_vendedor` - único no sistema; para este campo existe a palavra reservada "`generico`";
- `senha_vendedor` - a senha de acesso do vendedor ao sistema, armazenada em forma criptografada;
- `valido` - indica se o vendedor é ou não válido;
- `generico` - indica se o vendedor utiliza o sistema de micropagamentos genéricos ou o sistema de micropagamentos específicos;
- `IP` - o endereço IP do *site* do vendedor;
- `porta` - a porta TCP que o vendedor utiliza para as conexões das *servlets* (normalmente 8080);
- `nome` - o nome real do vendedor;
- `endereco` - o endereço do vendedor;
- `telefone` - o seu telefone.

Certificado Contém todos os certificados gerados no sistema, válidos ou não, expirados ou não, para os dois tipos de vendedores. Seus campos são:



Fraude

Column Name	Condensed Type	Nullable
nome_usuario	varchar(30)	NOT NULL
nome_vendedor	varchar(30)	NOT NULL
ficha	varchar(28)	NOT NULL
ordem	numeric(18, 0)	NOT NULL
valor	numeric(18, 0)	NOT NULL
data	datetime	NOT NULL
ip	varchar(15)	NOT NULL
tipo	numeric(18, 0)	NOT NULL
autorizacao	varchar(100)	NOT NULL

Historico

Column Name	Condensed Type	Nullable
nome_usuario	varchar(30)	NOT NULL
nome_vendedor	varchar(30)	NOT NULL
valor	numeric(18, 0)	NOT NULL
data	datetime	NOT NULL
autorizacao	varchar(100)	NOT NULL
ficha	varchar(28)	NOT NULL

Usuario

Column Name	Condensed Type	Nullable
nome_usuario	varchar(30)	NOT NULL
senha_usuario	varchar(30)	NOT NULL
valido	char(1)	NOT NULL
nome	varchar(50)	NULL
cpf	char(11)	NOT NULL
endereço	varchar(50)	NULL
telefone	varchar(15)	NULL

Certificado

Column Name	Condensed Type	Nullable
id_certificado	numeric(18, 0)	NOT NULL
nome_usuario	varchar(30)	NOT NULL
nome_vendedor	varchar(30)	NOT NULL
data_ger	datetime	NOT NULL
data_exp	datetime	NOT NULL
num_fichas	numeric(18, 0)	NOT NULL
valor_ficha	numeric(18, 0)	NOT NULL
generico	char(1)	NOT NULL
valido	char(1)	NOT NULL

Ficha

Column Name	Condensed Type	Nullable
id_certificado	numeric(18, 0)	NOT NULL
conteudo	varchar(28)	NOT NULL
ordem	numeric(18, 0)	NOT NULL
ja_gasta	char(1)	NOT NULL

Vendedor

Column Name	Condensed Type	Nullable
nome_vendedor	varchar(30)	NOT NULL
senha_vendedor	varchar(10)	NOT NULL
valido	char(1)	NOT NULL
generico	char(1)	NOT NULL
ip	varchar(15)	NULL
porta	numeric(18, 0)	NULL
nome	varchar(50)	NULL
endereco	varchar(50)	NULL
telefone	varchar(50)	NULL

Figura 5.1: Base de dados do agente.

- `id_certificado` - o número serial do certificado;
- `nome_usuario` - o nome do usuário ao qual pertence o certificado;
- `nome_vendedor` - o nome do vendedor ao qual pertence o certificado;
- `data_ger` - a data de geração do certificado;
- `data_exp` - a data de expiração do certificado; esse número pode ser alterado em um arquivo de configuração e seu valor padrão é de 1 ano após a data de geração do certificado;
- `num_fichas` - o número de fichas que o certificado contém;
- `valor_ficha` - o valor de cada ficha em centavos de Real;
- `generico` - indica se o certificado se aplica ao sistema de micropagamentos genéricos ou ao sistema de micropagamentos específicos;
- `valido` - indica se o certificado está válido ou não.

Ficha Contém todas as fichas, gastas ou não, geradas para todos os certificados da tabela Certificado. Seus campos são:

- `id_certificado` - indica a qual certificado a ficha pertence;
- `conteudo` - a ficha em si, ou seja, o valor gerado pela função de *hash*, codificado em Base64;
- `ordem` - o número de ordem da ficha;
- `ja_gasta` - indica se a ficha já foi gasta ou não do ponto de vista do agente, ou seja, se já foi recebido de algum vendedor o aviso de que essa ficha foi gasta. Esse campo é atualizado pela **ServAgHistorico** e serve também para controlar as fraudes e para gerar novos certificados.

Histórico Contém o histórico das transações bem-sucedidas (que não tenham sido consideradas fraudes) realizadas pelos vendedores e transmitidas ao agente. Sua função é a de geração de relatórios, além da manutenção geral do sistema. Seus campos são:

- `nome_usuario` - o nome do usuário que realizou a transação;

- `nome_vendedor` - o nome do vendedor que realizou a transação;
- `valor` - o valor da transação em centavos de Real;
- `data` - a data da transação, tal qual foi transmitida pelo vendedor;
- `autorizacao` - a autorização emitida pelo vendedor para essa transação;
- `ficha` - a ficha utilizada pelo usuário para essa transação.

Fraude Contém o histórico de fraudes detectadas pela **ServAgHistorico**. Tem a função de gerar relatórios e permitir o controle de fraudes por parte dos administradores do sistema. Seus campos são:

- `nome_usuario` - o nome do usuário envolvido com a fraude;
- `nome_vendedor` - o nome do vendedor envolvido com a fraude;
- `ficha` - a ficha sobre a qual foi detectada a fraude; pode ser uma ficha não existente na base de dados ou uma ficha que já havia sido gasta anteriormente;
- `ordem` - a ordem da ficha envolvida, tal qual informada pelo vendedor;
- `valor` - o valor da transação sobre a qual foi feita a fraude;
- `data` - a data da detecção da fraude, incluindo minutos e segundos;
- `ip` - o endereço IP do vendedor que informou a transação fraudulenta;
- `tipo` - indica o tipo de fraude: se por ficha não encontrada ou por ficha já gasta;
- `autorizacao` - a autorização emitida pelo vendedor para essa transação.

5.2.2 Vendedor

O módulo do vendedor é composto por três *servlets*, dois programas *standalone* e uma base de dados.

Programas

As *servlets* constituem o núcleo principal do módulo do vendedor; são as responsáveis pela interação direta do usuário com o sistema, no momento do primeiro contato e durante a transação de compra. São elas:

ServVendOpcaoUsuario Ao clicar em um produto pago na página do vendedor o seu navegador abre inicialmente um diálogo perguntando a ele se já é cadastrado no sistema. A resposta a essa pergunta é passada então à **ServVendOpcaoUsuario** juntamente com os seguintes parâmetros: o nome do produto escolhido e a URL da página em que a escolha foi feita.

A *servlet* pode fornecer duas respostas a essa requisição: se o usuário já for cadastrado, ela enviará uma página contendo a **AppletUsuario**, capaz de realizar a compra; se não for cadastrado, ela envia uma página de cadastro, que mais tarde enviará os dados para a **ServAgCadastro** fazer o processamento.

A URL da página que chamou a **ServVendOpcaoUsuario** é repassada por essa *servlet* e durante todo o trajeto de cadastramento do usuário para que ao final desse processo o seu navegador possa ser redirecionado de volta a essa página; isso permite que ele possa continuar o processo de compra do ponto em que parou para fazer o seu cadastramento e compra inicial de fichas.

ServVendFicha Essa é a principal *servlet* do módulo do vendedor. É a responsável por receber a ficha durante uma transação, verificar a sua autenticidade e emitir uma autorização. A entrada recebida é uma mensagem no formato XML contendo o nome do usuário, a ficha, sua ordem, o nome do produto, o valor da compra, o número serial do certificado e a ficha raiz assinada. Os passos de seu processamento são:

1. verifica se o número serial do certificado faz parte da tabela local de certificados revogados; em caso positivo, o processamento é abortado;
2. verifica a assinatura digital da ficha raiz com a chave pública do agente; se houver erro na verificação, aborta o processamento;
3. de posse do número de ordem k da ficha recebida e do número de ordem j da ficha raiz, calcula $k - j$ vezes a função de *hash* sobre a ficha e compara o resultado com a ficha raiz; se forem diferentes, retorna ao usuário um autorização de valor -1;
4. se as fichas forem iguais, gera um número aleatório de autorização e preenche

com esse número a tabela `Autorizacao` da base de dados, a ser descrita na página 59; ao mesmo tempo, envia para o usuário a autorização gerada;

5. se o sistema estiver configurado para o envio imediato de fichas ao agente, monta uma mensagem XML contendo o nome do vendedor, o nome do usuário, a ficha, seu número de ordem, a data da transação, o valor da compra e a autorização gerada e a envia para a **ServAgHistorico**;
6. atualiza a tabela `Historico` da base de dados local (descrita na página 59), inserindo os valores da transação e um campo informando se a ficha já foi enviada ao agente ou não.

ServVendProduto Essa *servlet* foi criada, de forma experimental, para receber a autorização e devolver o produto ao usuário. Os parâmetros que recebe são o nome do usuário, o nome do produto e a autorização. Sua função é consultar a tabela Autorização da base de dados do vendedor, verificar se as informações submetidas são autênticas e se a autorização não expirou. Em caso positivo, o produto é devolvido ao usuário. Em nossa implementação, isso é feito através da abertura de uma conexão com outra *servlet* que, por ser parte de outro projeto específico para um produto, não faz parte do escopo deste trabalho.

Fazem parte ainda do módulo do vendedor dois programas *standalone*, com funções auxiliares de automatização de comunicações:

VendHistorico Acorda a intervalos de tempo configuráveis e verifica se há, na tabela `Historico`, transações não informadas ao agente. Em caso positivo, monta um lote de informações em uma mensagem XML e o envia para **ServAgHistorico**. Recebe uma resposta dessa *servlet* e a partir dela atualiza a tabela `Historico` local, marcando o campo `ja_enviada` com valor "S";

VendedorCRL Acorda a intervalos de tempo configuráveis e faz uma pesquisa junto ao agente utilizando a **ServAgCRL**, recebendo como resposta uma lista de números seriais de certificados revogados em formato XML. Os números são então

inseridos na tabela `CRL`[2] da base de dados do vendedor. A pesquisa pode ser feita por usuário, data ou vendedor, de acordo com um arquivo de configuração local.

Base de dados

A base de dados do vendedor destina-se a armazenar: o histórico de transações realizadas, para efeito de controle de movimentação e de fraudes; os produtos disponíveis e as autorizações geradas, para realizar a integração com o sistema de vendas e entrega do produto, além da CRL atual obtida do agente. Suas tabelas são:

Produto Armazena as informações básicas sobre os produtos necessárias para a realização das transações. Seus campos são:

- `nome_produto` - o nome do produto, único em cada vendedor;
- `preco`- o preço do produto, em centavos de Real;
- `descricao` - campo auxiliar contendo a descrição do produto, ainda não utilizado.

Autorizacao Contém as autorizações geradas pela **ServVendFicha** para cada transação bem-sucedida. Seus campos são:

- `nome_produto` - identifica o produto ao qual a autorização se refere;
- `aut` - a autorização em si é um valor aleatório de 20 *bytes* codificado em Base64;
- `nome_usuario` - o nome do usuário autorizado a obter o produto;
- `data_exp` - a data de expiração da autorização; esse valor é configurável e definido na **ServVendFicha**; o padrão é de 5 minutos a partir da transação.

Historico Contém todas as informações relativas às transações realizadas pelo vendedor, além do controle do envio de informações para o agente. Seus campos são:

- `nome_usuario` - o nome do usuário que realizou a transação;

[2] *Certificate Revokation List*

- `ficha` - a ficha recebida e autenticada na transação;
- `ordem` - o número de ordem da ficha recebida;
- `data` - a data da realização da transação;
- `valor_compra` - o valor da compra realizada, em centavos de Real;
- `ja_enviada` - informa se a transação em questão já foi informada ao agente ou não;
- `aut` - a autorização emitida para essa transação.

CRL Armazena os números seriais dos certificados revogados pelo agente; é composta por um único campo, chamado serial, que contém o número serial de cada certificado revogado.

5.2.3 Usuário

O módulo do usuário é composto por duas *applets*. Não há base de dados, mas o certificado fica armazenado no computador local. Como o certificado está armazenado de forma critpografada, espera-se que somente as *applets* do sistema possam modificá-lo. São elas:

AppletCert Tem a função de instalar o certificado do usuário, gravando-o no disco local. Seus passos são:

1. obtém do servidor o arquivo que contém o certificado; esse arquivo está armazenado em um diretório temporário e seu nome é a concatenação de um número aleatório com o nome do usuário, seguido pela terminação `.scdsid`. O caminho completo para esse arquivo, incluindo seu nome, é passado como um parâmetro para a *applet*;
2. apaga o arquivo temporário do servidor, através de uma chamada a **ServAgApaga**, passando como parâmetro o nome do arquivo;
3. grava o arquivo no disco do computador local. O nome do arquivo é escolhido como sendo o nome do usuário, seguido pelo caracter "_", seguido pelo nome do vendedor e pela terminação `.scdsid`. No caso de o certificado

ser para o sistema de micropagamentos genéricos o nome fica sendo apenas o nome do usuário seguido pela terminação `.scdsid`. Assim, um certificado para o usuário `pedro` exclusivo para o vendedor `loja-c` ficaria assim: `pedro_loja-c.scdsid`; um certificado para o mesmo usuário aplicável ao sistema genérico ficaria simplesmente `pedro.scdsid`.

O diretório em que o certificado será gravado foi escolhido como sendo o padrão utilizado pela linguagem Java, ou seja, o diretório *home* de cada usuário. Assim, em sistemas Unix esse é o diretório `/usuario/home` e em sistemas Windows é o `\profiles\usuario\desktop`.

Um detalhe importante na gravação do certificado em discos locais é a segurança da linguagem Java. O modelo de segurança original da máquina virtual dos navegadores não permite que uma *applet* tenha acesso ao disco local. Para isso é preciso assinar digitalmente a *applet*, dando a ela permissões explícitas para leitura ou gravação. Como a forma dessa assinatura é proprietária, é preciso criar diferentes versões da *applet* para diferentes navegadores de forma que o funcionamento seja uniforme sobre todos eles;

4. redireciona o navegador para a página inicial da compra. supõe-se aqui que o usuário chegou a essa *applet* através de um processo de compra, iniciado na página de um vendedor, para o qual não possuía certificado ou fichas suficientes. Após a geração das fichas, ele deve ser redirecionado para o ponto em que parou para que possa realizar a compra que desejava.

AppletUsuario É um ponto central do sistema, sendo responsável por manipular o certificado do usuário, enviar fichas de compra e receber autorizações. Os passos de seu processamento são:

1. pede ao usuário que digite seu nome de usuário e senha;
2. busca no diretório padrão do usuário o certificado, utilizando para a busca o parâmetro `nome_de_vendedor` recebido e a regra de formação de nomes de arquivos explicada acima;

3. encontrado o arquivo, tenta decriptografá-lo utilizando os primeiros 16 *bytes* do *hash* da senha do usuário como chave;
4. se não foi possível encontrar ou decriptografar o arquivo, remete o navegador para a **ServAgPedeFichas**;
5. se a decriptografia foi bem-sucedida, verifica a integridade do arquivo através do campo de verificação. Esse campo contém o *hash* de todo o certificado, excluindo a ficha raiz, codificado em Base64, e foi adicionado como um passo de segurança a mais no controle de alterações sobre o certificado.
6. se o certificado estiver íntegro, verifica a sua validade através da comparação entre a data atual, recebida do servidor como um parâmetro, e a data de expiração impressa no certificado. Se a validade estiver vencida o navegador é redirecionado para a página de pedido de certificado em **ServAgPedeFichas**;
7. obtém o número de ordem da próxima ficha a ser enviada; o algoritmo que determina esse número leva em conta o número apontado no certificado como sendo a próxima ficha a ser gasta e adiciona a ele o valor da compra dividido pelo valor de cada ficha. Assim, se a próxima ficha a ser gasta é a 2, cada ficha vale 5 e compra a ser feita for de 15, será enviada ao vendedor a ficha 4, gastando assim três fichas (2, 3 e 4) que correspondem ao valor necessário para a compra.
8. monta e envia ao vendedor, em **ServVendFicha**, uma mensagem XML contendo:
   - o nome do usuário;
   - o número serial do certificado;
   - a ficha;
   - o número de ordem da ficha;
   - o nome do produto;
   - o valor da compra e
   - a ficha raiz assinada.

9. recebe a resposta do vendedor; se a autorização for -1 redireciona o navegador do usuário para uma página de erro;
10. se a autorização vier completa, atualiza o arquivo do certificado, passando o ponteiro para a próxima ficha ser gasta (em nosso exemplo, a ficha 5), recalculando o valor de verificação e criptografando tudo novamente com a chave obtida a partir da senha do usuário;
11. redireciona o navegador para **ServVendProduto**, incluindo na URL o nome do produto, o nome do usuário e o valor da autorização recebida.

5.3 Mensagens

Para o completo entendimento do sistema μ**P** é necessária a descrição dos seus componentes temporários, as mensagens. Entre elas incluímos o certificado que, embora não seja uma mensagem propriamente dita, tampouco é uma base de dados ou um programa. Nesta seção descreveremos detalhadamente e com exemplos as mensagens mais significativas do sistema.

Ficha

A ficha é uma seqüência de 20 *bytes*, codificada em Base64 para que possa ser tratada como uma seqüência de caracteres ASCII. Está sempre acompanhada por seu número de ordem, que a posiciona dentro de toda a seqüência de fichas emitidas para cada usuário. Um exemplo de ficha é

```
<ficha id='7'>Lc6xmGYMUbaSu4iz+A//9Xwz8Vo=</ficha>
```

No exemplo acima, o campo `id` representa a ordem da ficha, enquanto a seqüência entre os delimitadores `<ficha ...>` e `</ficha>` é a representação em ASCII (Base64) da ficha em si.

Um ponto importante a ressaltar é que o sistema μ**P** insere nas ficha o número de ordem inverso ao da seqüência de cálculo da função de *hash*. Isso faz com que a última ficha da seqüência tenha o número de ordem mais baixo, tornando crescente a ordem com que as fichas são gastas. Essa decisão foi tomada para tornar a numeração mais

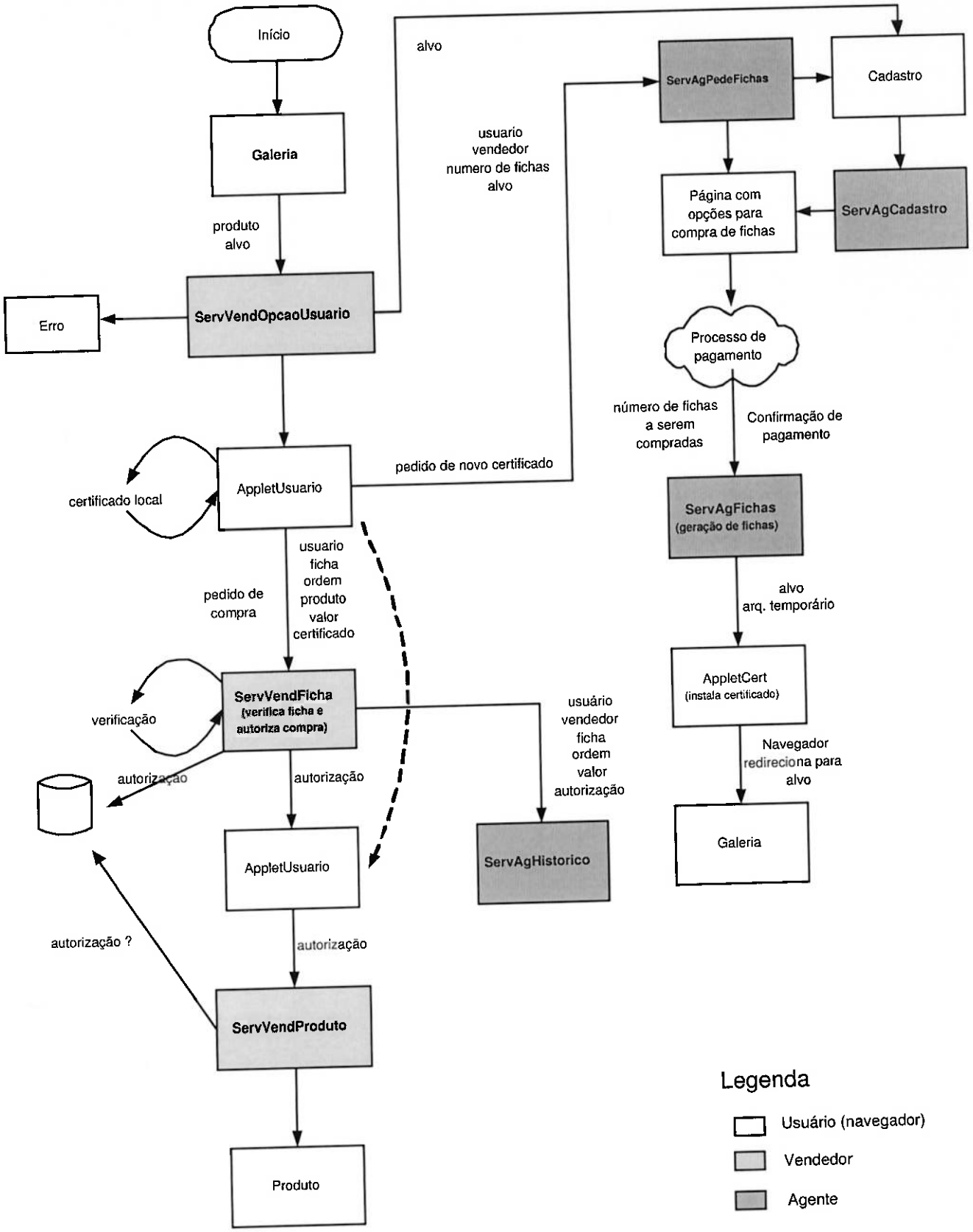


Figura 5.2: Interação entre os diversos objetos do μP

intuitiva, facilitando a manipulação das fichas e certificados e auxiliando a depuração dos programas.

Certificado

O certificado é armazenado no computador do usuário de forma criptografada e contém todas as informações necessárias para que a **AppletUsuario** realize uma transação de compra, incluindo a seqüência de fichas. Abaixo vemos um exemplo:

```
<uP>
    <serial>160</serial>
    <usuario>teste</usuario>
    <agente>LARC</agente>
    <vendedor>generico</vendedor>
    <data_exp>2003/06/25 20:19:03.119</data_exp>
    <fichas tam='97' valor='5' proxima='10'>
        <ficha id='100'>uuKll6l4RpWUOQEMFtC4zjHVuJI=</ficha>
        <ficha id='99'>LwOJMwisXq7vB91EUFp2gnwYu1Q=</ficha>
        .
        .
        .
        <ficha id='6'>uBYVxCZC6zRlyqsd/N6vUQQwdQk=</ficha>
        <ficha id='5'>c5vZxaR61Ah2fgxqZcvGMovlFOU=</ficha>
        <ficha id='4'>bkrBiqRs5x37tIeyje35J5AfNWO=</ficha>
    </fichas>
    <ver>xpyhQg8g3ojpAPcRzk8OCdPWTuM=</ver>
    <cert>PHVQPjxzZXJpYWw+MTYwPC9zZXJpYWw+PHVzdWFyaW8+dGVzdGU8L3V
    zdWFyaW8+PGFnZW50ZT5MQVJDPC9hZ2VudGU+PHZlbmRlZG9yPmdlbmVyaWNv
    PC92ZW5kZWRvcj48ZGF0YV9leHA+MjAwMy8wNi8yNSAyMDoxOTowMy4xMTk8L
    2RhdGFfZXhwPjxmaWNoYXMgdGFtPSc5NycgdmFsb3I9JzUnIHByb3hpbWE9Jz
    QnPjxmaWNoYSBpZD0nMyc+czZveEtqeThCWHFBbFEyb0dSSFVUQVpIMS9BPTw
    vZmljaGE+PC9maWNoYXM+PC91UD5nQXU5MXdOMkU1dzNyanV0c29jR3JLOEgz
```

```
    bUIyQ3pIcVJNKOR6NWMxYkJNM1N5TU55NFg4M2xRT3AxYOt2cmtkV2ZqalMzM
    1VTWWNudDBxROlpVm9YSWEOTEYvTU1yekVkZ3NQb1VvRGtTQXBIQUNkV2tLby
    t3TjMrbG1BbDBlQW5JNlJaTnVGOVBJUW1tTOJyNkFOYTRzQXZqTDNLZnQwZU4
    vRVo2dDhCZkk9</cert>
</uP>
```

No exemplo acima, os campos têm os seguintes significados:

`serial` - contém o número serial do certificado, seqüencial e único para todos os certificados gerados no sistema;

`usuario` - contém o nome de usuário do dono do certificado;

`agente` - contém o nome do agente emissor do certificado;

`vendedor` - contém o nome do vendedor (ou a palavra reservada "`generico`") ao qual o certificado se destina;

`data_exp` - contém a data de expiração do certificado;

`fichas` - contém a seqüência de fichas; o atributo `tam` indica o número de fichas presentes na seqüência, o atributo `valor` guarda o valor de cada ficha em centavos de Real e o atributo `proxima` indica qual a próxima ficha não gasta do certificado;

`ver` - contém o valor de verificação do certificado, que nada mais é que o *hash* de todo o conteúdo anterior ao delimitador `<ver>`, codificado em Base64.

`cert` - contém a ficha raiz assinada, que por razões históricas do desenvolvimento do projeto também pode ser conhecida como certificado do vendedor (ver capítulo 4). Como a ficha raiz está envolvida por uma mensagem XML com delimitadores iguais aos do certificado optou-se por codificar essa mensagem em Base64 para incluí-la no mesmo sem causar problemas para os interpretadores XML.

Ficha raiz

A ficha raiz é enviada ao vendedor dentro de um pequeno certificado, também conhecido como certificado do vendedor. Abaixo vemos um exemplo:

```
<uP>
<serial>160</serial>
<usuario>teste</usuario>
<agente>LARC</agente>
<vendedor>generico</vendedor>
<data_exp>2003/06/25 20:19:03.119</data_exp>
<fichas tam='97' valor='5' proxima='4'>
    <ficha id='3'>s6oxKjy8BXqAlQ2oGRHUTAZH1/A=</ficha>
</fichas>
</uP>
gAu91wt2E5w3rjutsocGrK8H3mB2CzHqRM+Dz5c1bBM3SyMNy4X831QOp1cKv
rkdWfjjS33USYcnt0qGIiVoXIa4LF/MMrzEdgsPoUoDkSApHACdWkKo+wN3+1
mAl0eAnI6RZNuF9PIQmmOBr6ANa4sAvjL3Kft0eN/EZ6t8BfI=
```

O certificado do agente contém os mesmos campos iniciais do certificado do usuário e pode ser utilizado pelo vendedor para autenticar os dados recebidos junto com a ficha. O certificado indica o mesmo número de fichas do certificado do usuário, porém só contém uma ficha na seqüência. Como se pode observar pelo exemplo acima, a ficha contida nesse certificado é de ordem 3, exatamente um número abaixo da ficha mais baixa do certificado de usuário mostrado na página 65.

O campo que segue o delimitador </uP> é a assinatura digital criada pelo agente sobre o certificado de vendedor que a antecede, que pode ser verificada com a chave pública do agente nos vendedores; essa verificação é feita na **ServVendFicha**.

Pedido de compra

O pedido de compra é uma mensagem em formato XML, enviada da **AppletUsuario** para a **ServVendFicha** no momento da transação. Vejamos o exemplo abaixo:

```
<compra>
    <usuario>teste</usuario>
    <serial>224</serial>
    <ficha>11qp1s9qg07WUEwO6GF4DSYssts=</ficha>
```

```
    <ordem>48</ordem>
    <produto>Lena.jpg</produto>
    <valor>15</valor>
    <cert>PHVQPjxzZXJpYWw+MjIOPC9zZXJpYWw+PHVzdWFyaW8+dGVzdGU8L3V
    zdWFyaW8+PGFnZW5OZT5MQVJDPC9hZ2VudGU+PHZlbmRlZG9yPmdlbmVyaWNv
    PC92ZW5kZWRvcj48ZGF0YV9leHA+MjAwMy8wOC8xNSAxNDowMjoyNy42NDk8L
    2RhdGFfZXhwPjxmaWNoYXMgdGFtPScxNzAnIHZhbG9yPSc1JyBwcm94aW1hPS
    czMSc+PGZpY2hhIGlkPSczMCc+QOpIMFgOYVZoVTRYaVBhY2dGdzdFcmlPK2Y
    OPTwvZmljaGE+PC9maWNoYXM+PC91UD5KSjRWcWV1RGpsYnlnVlNwSVQvRWl2
    M2x1RlpTdEkOM1p1Zlhkc DhFKOoxR1VnNlhpSUtyWVFkMjJndmZLNUxQS1AvR
    zhnYW84YOdmZVlKdjAzSU4rcy8wVmh3QUdaSmJjUzJIUkxIaS8rd2w2ak4weX
    hQTS9KcEFIUFFGeUMxTGNHU1dIZVMwZlpiTWhYam9ZMFB6eFRBaUtHQUJNVFI
    OckZSdEtLMOwrdUU9</cert>
</compra>
```

Seus campos são explicados a seguir:

`usuario` - contém o nome de usuário do usuário que está enviando a ficha;

`serial` - contém o número serial do certificado ao qual a ficha pertence;

`ficha` - contém a ficha que autentica a transação;

`ordem` - contém a ordem da ficha que está sendo enviada;

`produto` - contém o nome do produto que está sendo comprado;

`valor` - contém o valor da transação; é um indicativo de quantas fichas estão sendo enviadas;

`cert` - é o certificado do vendedor, que contém a ficha raiz, codificado em Base64, tal qual é encontrado no certificado do usuário.

Autorização

A autorização é um número aleatório de 20 *bytes*, codificado em Base64 para poder ser tratado como uma seqüência de caracteres ASCII. Ela é gerada por **ServVendFicha** e

enviada de volta para a **AppletUsuario** dentro do cabeçalho HTTP, como vemos no exemplo abaixo:

```
HTTP/1.0 200 OK
Date: Thu, 15 Aug 2002 17:02:38 GMT
Server: Tomcat Web Server/3.3.1 Final ( JSP 1.1; Servlet 2.2 )

aut=t3XWE+x3F++iRV2XUn5VW3uaKQg=
```

Histórico

A mensagem a que chamamos de histórico é aquela que informa o agente sobre o gasto de uma ou mais fichas no vendedor. Essa mensagem é utilizada para atualizar o histórico do agente, de modo que ele possa realizar os pagamentos necessários aos vendedores e detectar possíveis fraudes ocorridas no sistema. Vejamos o exemplo abaixo:

```
<historico>
    <vendedor>generico</vendedor>
    <fichas>
        <ficha usuario='teste' conteudo='TVPMeoSRN6yItbg/hx3SgFkcYoE='
        ordem='51' data=' 2002/08/19 16:21:13.997' valor_compra='15'
        aut='sORhtGGMvFd8vlHeN4uojFjYnsU='/>
    </fichas>
    <tam>1</tam>
</historico>
```

Podemos observar a presença de três campos:

`vendedor` - indica ao agente quem é o vendedor realizando a transmissão de informação. Caso esteja preenchido com genérico, o agente terá que utilizar o endereço IP do vendedor para identificá-lo;

`fichas` - Contém a coleção de fichas (no exemplo apenas uma) que estão sendo relatadas ao agente nesse lote;

`tam` - contém o número de fichas que estão sendo informadas nesse lote.

O campo `ficha`, dentro do campo `fichas`, possui uma série de atributos relativos a cada ficha informada:

`usuario` - o nome do usuário que gastou a ficha;

`conteudo` - a ficha gasta;

`ordem` - a ordem da ficha;

`data` - a data em que foi realizada a transação;

`valor_compra` - o valor da transação;

`aut` - a autorização emitida pelo vendedor para a transação.

5.4 Considerações finais

A implementação do sistema μ**P** , descrita neste capítulo, apresenta uma característica fundamental: a flexibilidade. A divisão do sistema em módulos permite que os papéis de cada pedaço de programa sejam claramente definidos e portanto mais facilmente modificados e corrigidos. A utilização de uma base de dados centralizada em cada módulo leva a uma grande maleabilidade desses módulos, permitindo trocas simples e eficientes de dados entre cada um dos programas que os compõem.

O sistema apresenta, além de flexibilidade de implementação, uma grande flexibilidade funcional: permite que se utilize micropagamentos específicos ou genéricos, que se informe transações ao agente imediatamente ou em lotes e permite também que os usuários percam seus certificados ou mudem de computador sem perder o dinheiro já investido.

No próximo capítulo serão apresentados os resultados da execução dos programas que compõem a implementação, além de uma avaliação analítica do sistema.

Capítulo 6

Resultados Obtidos

O sistema μP será avaliado segundo dois aspectos: o de sua funcionalidade e o analítico. A avaliação de funcionalidade se dará seguindo um plano de testes, enquanto que a avaliação analítica seguirá uma estrutura para avaliação de sistemas de micropagamentos proposta em [20].

6.1 Avaliação funcional

Para avaliar a funcionalidade do sistema foram definidos os seguintes testes:

- compra normal;
- erro de senha;
- compra com número insuficiente de fichas;
- compra sem certificado;
- compra com ficha inválida;
- fraudes;

Para o ambiente de testes foi cadastrado apenas um vendedor no sistema, configurado para funcionar no sistema de micropagamentos genéricos. Foi criada uma página para esse vendedor, a que chamamos Galeria, na qual aparecem três imagens pela visualização das quais o usuário supostamente tem que pagar alguma quantia. Entre as três imagens, apenas uma foi cadastrada como produto e seu preço fixado em 15 centavos. Como o valor das fichas foi definido em 5 centavos, a visualização dessa imagem custa três fichas ao usuário.

Compra Normal

O resultado de uma compra normal é o primeiro objetivo na implementação do sistema. É também o resultado em que se percebe mais facilmente qualquer problema no funcionamento de algum módulo. É natural, portanto, que esse teste tenha quase sempre um resultado positivo, e que quando isso não acontece os programas sejam rapidamente corrigidos.

Nosso teste de normalidade consiste em, a partir do zero, como um novo usuário do sistema, realizar os seguintes passos:

1. o usuário acessa a Galeria do vendedor e clica na imagem que quer ver;
2. o seu navegador abre um diálogo perguntando se o usuário já está cadastrado no sistema;
3. o usuário responde Não;
4. o navegador é redirecionado para a página de cadastro, mostrada na figura 6.1.
5. após realizar o cadastro, o vendedor é remetido para a página de compra de fichas, em que pode decidir pelo número inicial de fichas que deseja adquirir, como pode ser visto na figura 6.2;
6. ao selecionar esse número o usuário recebe uma página com a **AppletCert**; antes de ser executada, ela pede que o usuário autorize o seu acesso ao disco, como pode ser visto na figura 6.3; após a aceitação, a *applet* grava no disco local um certificado criptografado, que pode ser visto na figura 6.4;
7. o navegador do usuário é então redirecionado para a Galeria do vendedor, onde ele clica novamente na imagem desejada;
8. o navegador pergunta novamente se o usuário já está cadastrado no sistema, e desta vez a resposta é Sim;
9. o usuário recebe então uma página contendo a **AppletUsuario**, que inicia sua execução abrindo um diálogo em que pede nome de usuário e senha, como pode ser visto na figura 6.5;

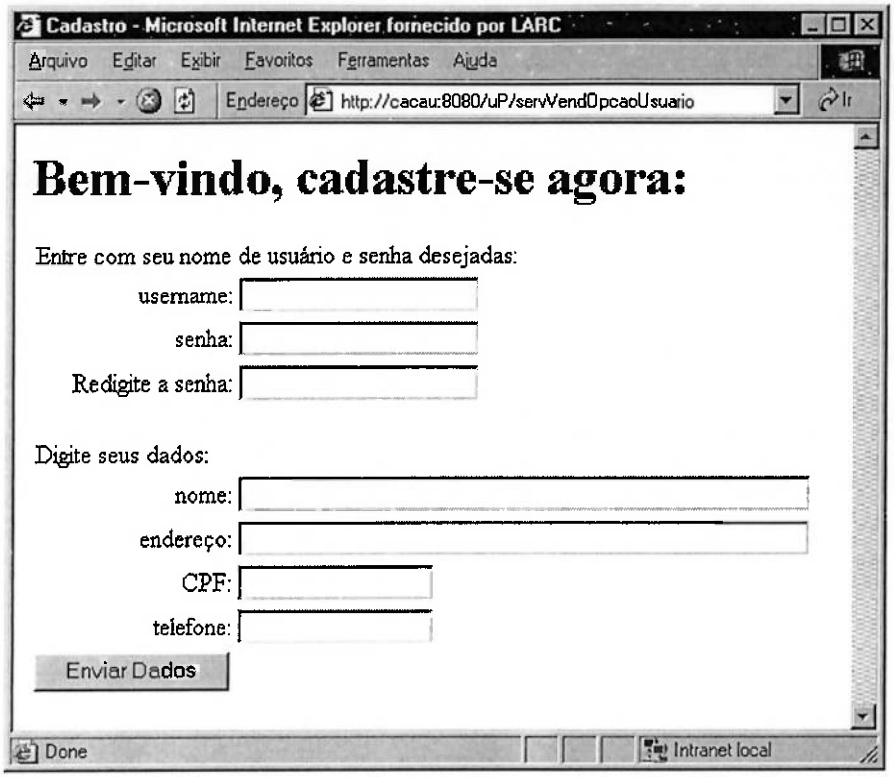


Figura 6.1: Tela de cadastro

10. a partir desse momento todo o processo da transação corre automaticamente, e o usuário é imediatamente remetido para a página do produto, que pode ser vista na figura 6.6.

Erro de senha

A senha do usuário é pedida a ele em três momentos: no de seu cadastro, no momento de geração de novo certificado e no momento de uma compra. O erro de senha só faz sentido nos dois últimos, uma vez que no cadastro a senha está sendo criada e só pode ser comparada com ela mesma, repetida em dois campos.

Quando o usuário digita uma senha errada no momento da geração de um novo certificado seu navegador recebe uma página informando o erro, que pode ser vista na figura 6.7.

Quando o usuário digita uma senha errada na **AppletUsuario**, a senha é pedida novamente por duas vezes. Se a terceira senha continuar errada, o usuário é automa-

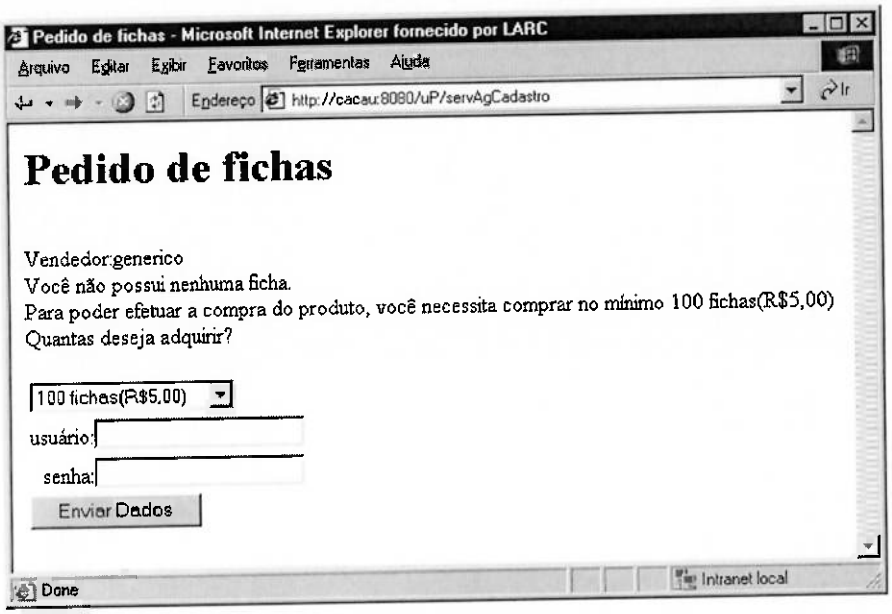


Figura 6.2: Pedido de fichas

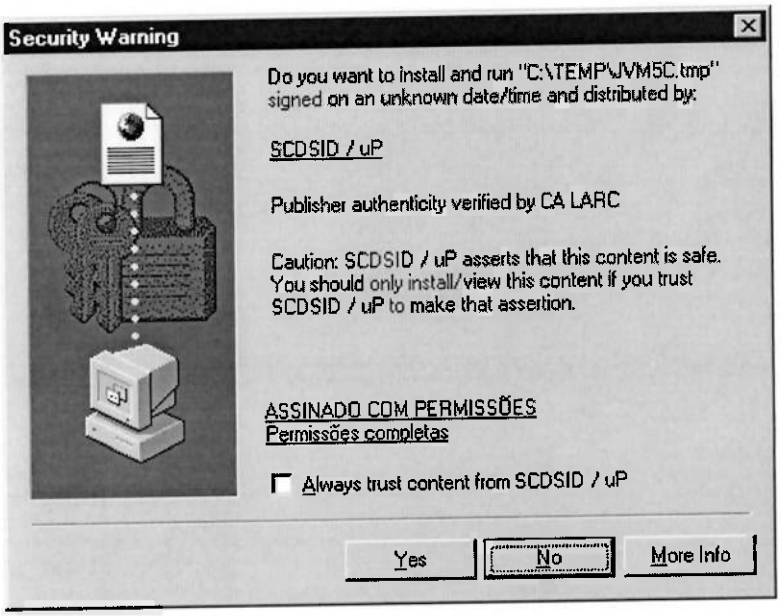


Figura 6.3: Aviso de segurança da **AppletUsuario**

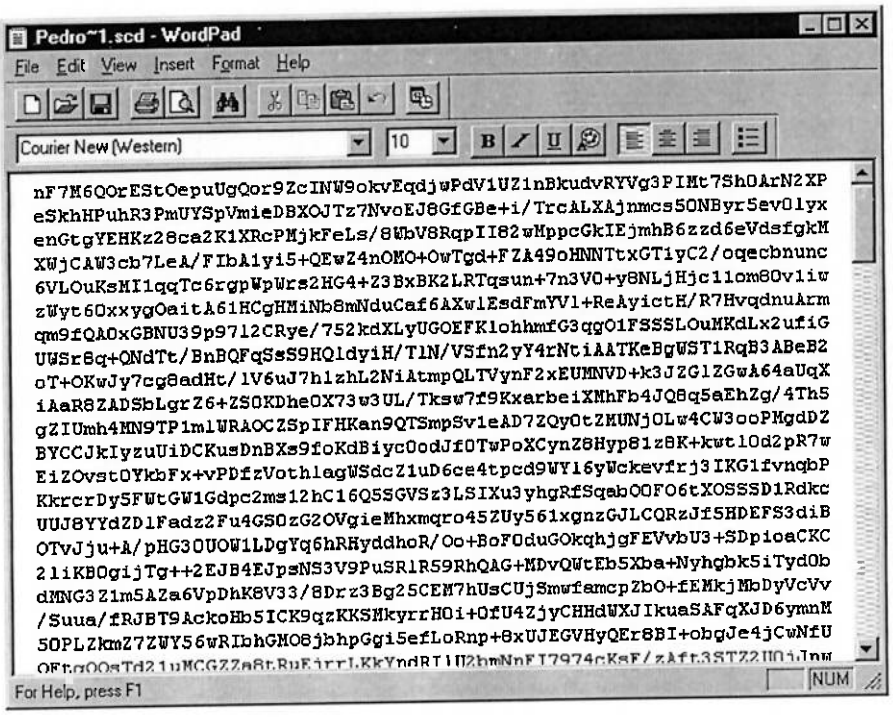


Figura 6.4: Certificado criptografado

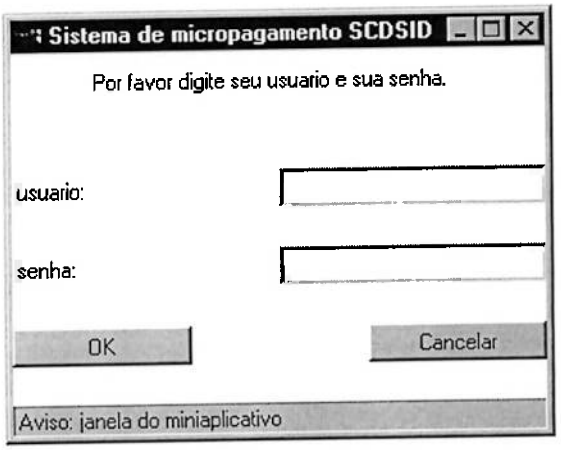


Figura 6.5: Diálogo apresentado por **AppletUsuario**

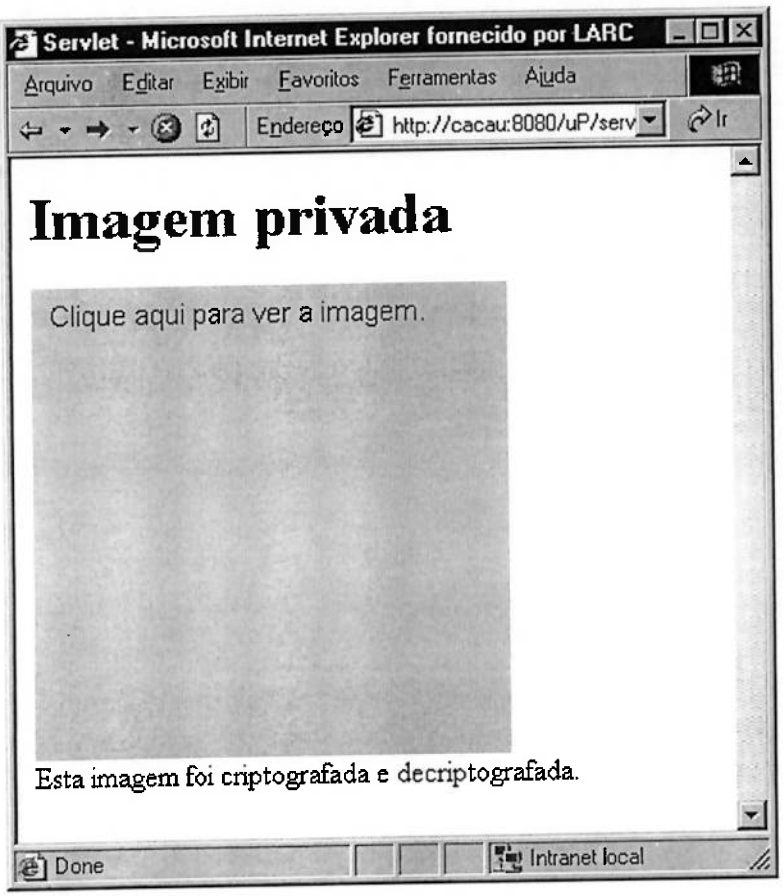


Figura 6.6: Página do produto

ticamente redirecionado para a página de geração de novo certificado. Considera-se nesse caso que o nome de usuário digitado estava certo e que o certificado foi corrompido e deve ser gerado novamente. Existe também a possibilidade de um usuário mal intencionado tentar acessar o certificado de outro usuário, sem saber a sua senha. Ele será redirecionado para a página de geração de certificados que, por sua vez, pede novamente a senha, e não permite a geração de certificado com a senha errada. Dessa forma, o usuário original continua protegido do ataque a seu certificado.

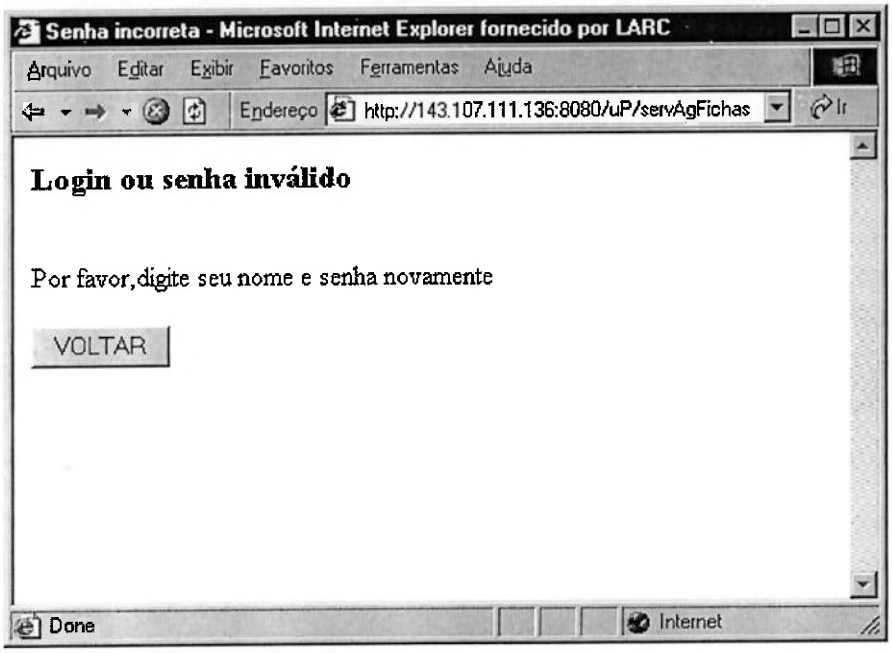


Figura 6.7: Aviso de senha errada.

Compra com número insuficiente de fichas

Quando o produto a ser comprado requer mais fichas do que o usuário possui em seu certificado lhe é enviada a página de pedido de novo certificado, que o informa sobre a quantidade de fichas necessária para fazer a compra e dá as opções de compra de acordo com os valores mínimos configurados no sistema.

Para realizar esse teste modificamos o preço de nosso produto único na base de dados, de forma que custasse o valor de 5 fichas a mais que o número possuído pelo usuário. O resultado pode ser visto na figura 6.8.

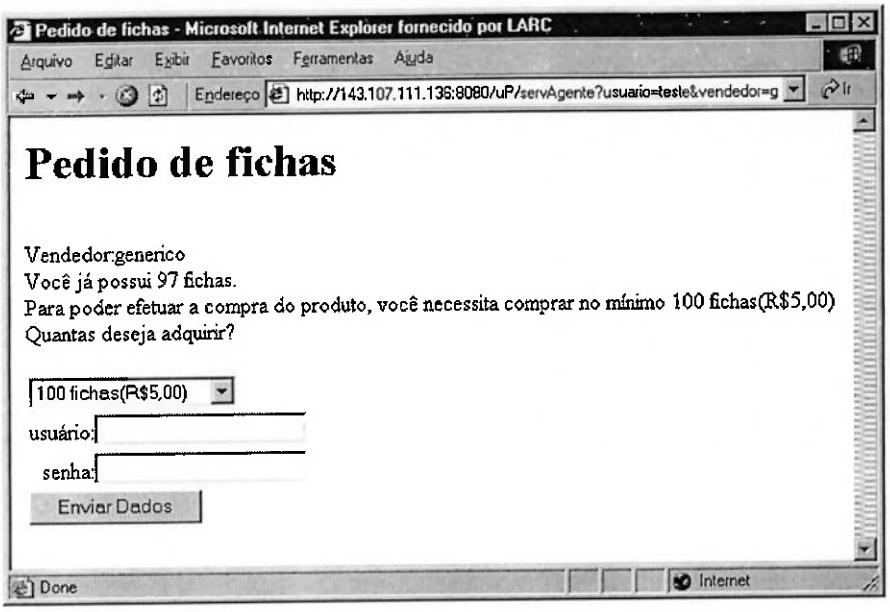


Figura 6.8: Número insuficiente de fichas.

Compra sem certificado

Quando o certificado do usuário não está presente no diretório padrão do disco local a **AppletUsuario**, após obter do diálogo o nome do usuário e tentar encontrar o arquivo com o nome correspondente, redireciona o navegador para a página de pedido de certificado, na qual o usuário será informado do ocorrido e poderá fazer o pedido de um novo certificado, sem ter necessariamente que comprar novas fichas. A resposta a essa situação pode ser vista na figura 6.9.

Compra com ficha inválida

Essa situação simula um tipo possível de fraude: o envio ao vendedor de uma ficha forjada fora do sistema. Como a verificação de autenticidade da ficha dá resultado negativo, a resposta do vendedor é imediata com uma autorização de valor -1. O usuário é redirecionado então para uma página que o informa do erro, como visto na figura 6.10.

Fraudes

Existe a possibilidade de que um usuário mal intencionado e conhecedor da tecnologia utilizada construa programas para manipular seu certificado e enviar fichas forjadas

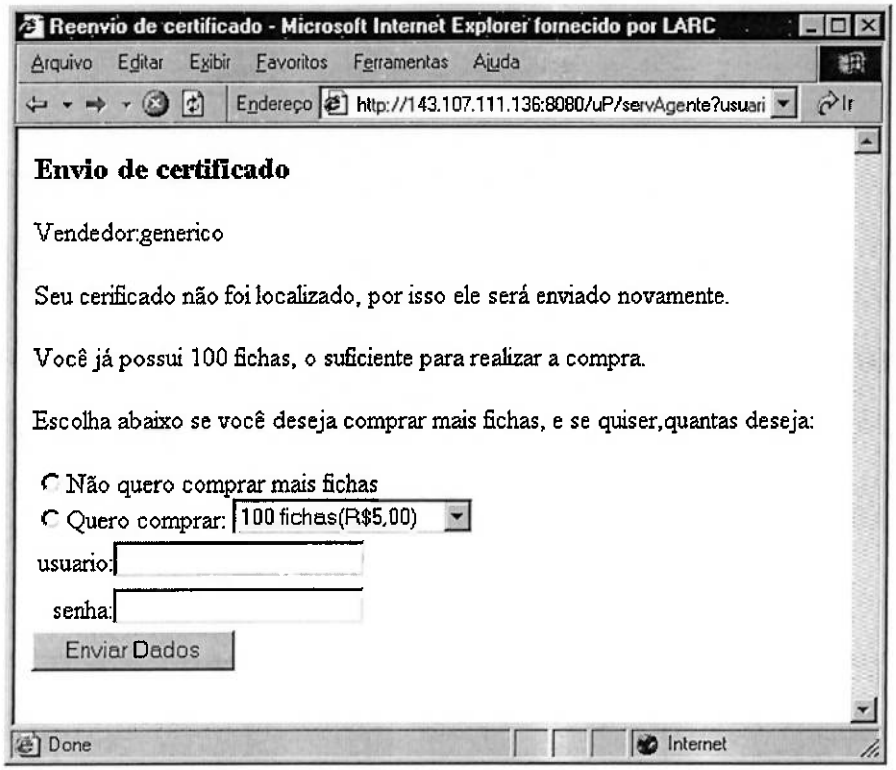


Figura 6.9: Resposta do sistema a um certificado perdido.

Figura 6.10: Resposta ao envio de autorização inválida.

para os vendedores. Esse usuário poderia também manipular o ponteiro para a próxima ficha e gastar apenas uma ficha para realizar uma compra em que seriam necessárias dez, ou até reenviar a mesma ficha várias vezes para um ou mais vendedores.

Como o esquema adotado é totalmente distribuído no momento da transação, os vendedores não têm como se precaver contra esse tipo de fraude. Em nossos testes, no entanto, elas são detectadas imediatamente pelo agente, no momento em que este é informado sobre os gastos de fichas. Outra possibilidade, ainda não implementada, é a de o agente fazer um simples controle contábil dos gastos de cada usuário e revogar o seu certificado se em algum momento um usuário gastar mais do que já comprou em fichas.

As fraudes realizadas por vendedores, no entanto, são mais difíceis de detectar. Um vendedor mal intencionado poderia manipular os programas e bancos de dados e enviar ao agente fichas aparentemente autênticas que na verdade foram gastos em outro vendedor. Isso é possível porque, de posse de uma ficha, basta ao vendedor aplicar a função de *hash* sobre ela algumas vezes para obter todas as fichas já gastas antes dela, mesmo que essas fichas tenham sido gastas em outro vendedor. Essa fraude apareceria no sistema μP como uma fraude de usuário.

A solução para esse problema, também não implementada, é fazer uma análise da ocorrência de fraudes e tentar descobrir, através de datas e horários informados para as transações, quais foram feitas pelos vendedores e quais foram feitas por usuários. Outra possibilidade é analisar os padrões de ocorrência de fraudes e a partir daí suspeitar de uma possível fraude de vendedor. Em princípio, no entanto, e na implementação experimental que fizemos, os vendedores são considerados confiáveis pelo sistema.

6.2 Avaliação analítica

A avaliação analítica do μP será feita de acordo com a estrutura proposta em [20]. Essa estrutura prevê uma avaliação qualitativa de nove parâmetros, separados em três dimensões: a dimensão microeconômica, a dimensão tecnológica e a dimensão social.

Na dimensão microeconômica os autores levantaram três parâmetros: baixo custo de transação, transações atômicas e a base de usuários. Os custos de transação são

um parâmetro muito difícil de avaliar, pois envolvem um estudo detalhado de todos os custos fixos e marginais envolvidos na implantação de um sistema; assim, serão considerados apenas os custos de computação e comunicação envolvidos em cada transação. Transações atômicas são um requisito importante de funcionamento do sistema, e significam que os pagamentos têm que ser completos; a troca de bens por pagamentos também deve acontecer, ou seja, não se pode dar pagamento sem receber um bem e não se pode receber um bem sem dar pagamento. A base de usuários, por fim, representa o alcance econômico do sistema; quanto maior o número de pessoas que utilizarem o sistema, mais útil ele será para essas pessoas.

Na dimensão tecnológica foram levantados quatro parâmetros: segurança, confiabilidade, escalabilidade e latência. Segurança implica em confidencialidade, integridade e autenticação de mensagens, pois os sistemas funcionam em redes abertas. Confiabilidade depende da disponibilidade do sistema ou do risco de uma tentativa de utilização do sistema em determinado momento ser frustrada por razões intrínsecas ao sistema. Escalabilidade é a capacidade de crescimento do sistema, mantendo os outros requisitos nos mesmos patamares. Latência é o atraso apresentado pelo sistema em cada transação, que deve ser mantido pequeno mesmo em horários de pico.

Na dimensão social foram considerados dois parâmetros: pagamentos inter-usuários (ou *peer-to-peer*) e o grau de anonimidade oferecido pelo sistema. Os pagamentos inter-usuários são importantes para que o sistema de pagamentos proposto possa atingir um grau maior de aceitação social, pois uma economia real não funciona somente entre pessoas e estabelecimentos comerciais. O grau de anonimidade é importante para que o sistema tenha maior aceitação do público, embora isso seja, segundo os autores, controverso e discutível.

Para a avaliação os autores propõem o estudo detalhado do sistema em questão, principalmente no que se refere ao processo de transação de compra, e a atribuição de conceitos que variam entre "--" para requisito não atendido, "-" para requisito mal atendido, "o" para não decidido, "+" para requisito bem atendido e "++" para requisito muito bem atendido.

Passaremos agora à avaliação do μ**P** segundo esses critérios:

Baixo custo de transação

Quando consideramos uma transação, podemos tranqüilamente ignorar o envio de informações sobre essa transação ao agente e nos concentrarmos apenas no que concerne à troca de informações entre usuário e vendedor. Fazemos isso porque a troca de informações se dá após a conclusão da transação e pode ser configurada para ser feita de forma independente em lotes.

Assim, o custo de uma transação, em termos computacionais, é o custo de uma verificação de assinatura digital e de i aplicações da função de *hash*, onde i é a diferença entre o número de ordem k da ficha recebida e o número de ordem j da ficha raiz. O número de aplicações da função de *hash* é portanto variável, fazendo com que o custo de cada transação não seja determinístico. Mais ainda, esse número pode variar com o próprio tamanho do certificado que o usuário comprou.

Apesar disso, a aplicação de uma função de *hash* é muito rápida, mesmo repetida muitas vezes. Um motivo para atrasar a transação, no entanto, é a verificação da assinatura digital. Por esse motivo, avaliamos o custo de transação de nosso sistema com um conceito "+".

Transações atômicas

O sistema μ**P** foi desenhado e implementado de forma que a ficha não seja gasta sem que a autorização seja recebida. Se a **AppletUsuario** receber uma autorização igual a -1 ou não receber uma autorização, ela não atualizará o certificado do usuário, mantendo o estado da ficha como não gasta. A atomicidade da transação, no entanto, não prevê a entrega final do produto, que pode não acontecer por outros motivos, como por exemplo uma queda abrupta de conexão. A transação do μ**P** acaba no momento da recepção da autorização. Assim, podemos considerar que apenas uma parte da transação é atômica, o que resulta em um conceito "o".

Base de usuários

A base de usuários do μ**P** ainda não existe e não pode ser avaliada. O conceito adequado é, portanto, "o".

Segurança

O sistema μ**P** não apresenta, na implementação experimental, recursos para prover autenticidade, integridade ou confidencialidade para as mensagens trocadas entre os módulos. A decisão de não prover esse recurso foi tomada para simplificar a implementação funcional do sistema, deixando para desenvolvimentos futuros e para outros protocolos (como SSL, por exemplo) a tarefa de prover esses recursos.

O controle de fraudes do sistema é ainda falho e depende de atuação humana para a detecção de alguns tipos de fraudes, como por exemplo a geração de fichas falsas por vendedores. Embora esse modelo tenha sido seguido para atingir uma solução de compromisso entre custo de implementação e segurança, a sua avaliação continua sendo negativa: o conceito atribuído é "–".

Confiabilidade

Como a transação é verificada apenas no vendedor, a confiabilidade do sistema fica distribuída entre os vários vendedores participantes. Alguns problemas podem ocorrer quanto à utilização de *applets* Java nos computadores dos usuários, mas tomaremos por base a situação em que elas são autorizadas e funcionam normalmente. Um ponto central de falhas surge no momento da compra de fichas, que no entanto não faz parte da transação de compra propriamente dita. Assim, vamos considerar o sistema como semi-distribuído e atribuir a ele um conceito "+".

Escalabilidade

Até este momento não foi possível realizar testes de escalabilidade do sistema implementado. Consideraremos então a escalabilidade teórica, que se baseia no fato de o agente centralizar todas as operações, exceto a transação. Assim, embora no aspecto da compra o sistema seja distribuído, ele é bastante concentrado no que tange à administração.

A base de dados do agente deve conter todas as fichas geradas para todos os certificados de todos os usuários para todos os vendedores (embora consideremos que a maioria dos vendedores aceitará o sistema de micropagamentos genéricos), além do

histórico de todas as transações realizadas. A dependência de uma base de dados assim centralizada, que só pode ser acessada através de um único servidor, depõe contra a escalabilidade do sistema, deixando-o com um conceito "-".

Latência

A latência da transação é o tempo necessário para a verificação de uma assinatura digital e de um número variável de aplicações da função de *hash*, como descrito no item **Baixos custos de transação**. Embora seja um componente de atraso, a assinatura digital é, com a tecnologia atual, verificada em tempo razoável ([10]), embora muito maior que as aplicações de funções de *hash*. Assim, a latência média do **μP** é determinada por essa verificação, o que resulta em um conceito "+".

Pagamentos inter-usuários

O **μP** não apresenta a menor possibilidade de implantação de uma forma de pagamento inter-usuários. Pela forma como foi implementado, existe a necessidade de instalação de um conjunto de *servlets* e de uma base de dados para receber pagamentos, o que é inviável para esse tipo de pagamento. Assim, o conceito atribuído para esse requisito é "não se aplica".

Anonimidade

Da forma como está implementado, o sistema não tem nenhum grau de anonimidade: o usuário tem que se cadastrar, fornecendo até o seu CPF, para poder comprar as fichas. Existe, no entanto, uma possibilidade de implementação de anonimidade através da venda de cartões pré-pagos com nomes de usuário e senha pré-definidos para cada cartão. Embora seja um trabalho futuro, acreditamos que essa possibilidade pode ser um interessante valor a ser acrescentado ao sistema, e que eleva em um ponto o seu conceito quanto a esse quesito, que fica então sendo "-".

Resultados

Os resultados da avaliação analítica do sistema μ**P** podem ser vistos na tabela 6.1. Uma tabela semelhante é apresentada em [20] para os sistemas DigiCash, First Virtual e SET. Uma comparação entre os conceitos atribuídos para esses trabalhos pelos autores e os conceitos aqui atribuídos ao μ**P** nos leva a uma avaliação extremamente positiva deste trabalho, posicionando-o como uma alternativa viável para sistemas desse tipo.

	Requisito	μ**P**
Microeconômicos	Baixos custos de transação	+
	Transações atômicas	o
	Base de usuários	o
Tecnológicos	Segurança	-
	Confiabilidade	+
	Escalabilidade	-
	Latência	+
Sociais	pagamentos inter-usuários	não se aplica
	Anonimidade	-

Tabela 6.1: Avaliação final do μ**P**

Capítulo 7

Conclusão

O comércio eletrônico brasileiro está em fase de franca expansão, apesar dos percalços econômicos gerados por instabilidades externas e internas. Essa expansão leva a novas possibilidades de negócios, entre eles uma tendência intrínseca que leva à venda de informação em formato digital. Esse modelo de negócio apresenta características peculiares que requerem a implantação de um novo sistema de pagamentos que atenda aos requisitos de baixo custo e baixa latência.

Os sistemas de micropagamentos surgiram como resposta a essa tendência em países desenvolvidos e são agora uma aposta em países em desenvolvimento como o Brasil, em função da maturidade atingida pela nossa economia digital. Sistemas de micropagamentos apresentam as características necessárias para transações de baixos valores através de operações simples e razoavelmente seguras. O modelo de segurança de sistemas desse tipo é simplificado, representando uma solução de compromisso entre os custos de implementação e o nível de segurança necessário para as transações que realiza.

O μP é um sistema de micropagamentos experimental e totalmente funcional, desenvolvido a partir do sistema PayWord proposto em [18]. Sua característica principal é o fato de utilizar uma moeda eletrônica gerada em uma instituição central mas verificada de forma simples em outras localidades através do uso mínimo de operações computacionalmente intensas. O μP se diferencia por utilizar apenas um conjunto de fichas que permitem realizar compras em qualquer vendedor cadastrado. Sua interface Web desenvolvida em Java possibilita uma distribuição ampla através dos diversos sistemas que compõem o universo de clientes e vendedores da Internet, realizando nessas máquinas um mínimo de operações computacionalmente intensas.

Embora fosse inicialmente baseado em um sistema existente, o μP se aproveitou de idéias propostas em muitos sistemas e se mostrou, ao final do processo de criação e desenvolvimento, como um sistema novo e único, apresentando vantagens consideráveis so-

bre outros sistemas propostos, especialmente no que concerne à sua potencial adaptação à realidade do comércio eletrônico no Brasil.

7.1 Trabalhos Futuros

Existem dois aspectos para os quais podemos conceber futuros desenvolvimentos do μ**P** : aperfeiçoamentos do próprio sistema, especialmente em sua implementação, e o desenvolvimento de novas aplicações e modelos de negócio associados a sistemas de micropagamentos.

Sistema

Com relação ao sistema μ**P** , existem três propostas de melhoramento da implementação existente:

1. implementação de conexões seguras em pontos críticos de troca de dados; esse ponto essencial não foi implementado em razão da estratégia de implementação em camadas adotada, mas é necessário para a distribuição do μ**P** no ambiente da Internet; sua implementação, contudo, pode ser feita de forma simples e direta;

2. implementação de interface para obtenção de relatórios no agente e no vendedor; essa é uma funcionalidade básica de qualquer sistema computacional que foi deixada de lado para simplificar a implementação do sistema, mas é fundamental para a manutenção de um sistema operando em ambiente de produção;

3. melhoria das defesas contra fraudes; existem várias formas de melhorar as defesas contra fraudes do μ**P** sem perder suas características de latência e custo. Uma proposta é implementar um esquema de contabilidade simples, comparando, para cada usuário, o que foi comprado com o que foi gasto.

Aplicações

Entre as inúmeras idéias de novas aplicações que surgem a partir do desenvolvimento de sistemas de micropagamentos, destacamos três:

- a implementação de sistema anônimo com venda de cartões pré-pagos no varejo; cada cartão conteria um nome de usuário aleatório e uma senha, escondidos sob uma proteção que poderia ser eliminada através de raspagem. Esses usuários especiais já estariam pré-cadastrados no sistema e teriam uma quantidade de fichas já pagas. Isso possibilitaria que pessoas sem acesso a cartão de crédito (necessário no sistema atual para a aquisição imediata de fichas) pudessem utilizar o sistema μ**P** para fazer compras em computadores públicos, além de garantir um grau maior de anonimidade em função da dispensa do cadastro inicial e do processo de compra das fichas;

- o uso de fichas eletrônicas de micropagamentos em *smart cards* como forma de autorização para compras já efetuadas com outros sistemas de pagamento; o funcionamento seria o seguinte: imaginemos um usuário que quer comprar pela Internet duas entradas para ir ao cinema; esse usuário pode fazer isso hoje através de seu cartão de crédito, e isso não mudaria; a questão é que, depois de efetuada a compra, o usuário tem que imprimir o ingresso em casa ou retirá-lo na bilheteria, o que é feito através de um número de autorização recebido durante a compra; é nesse ponto que entra a ficha eletrônica: ao invés de receber um número simples, o usuário recebe uma ficha de micropagamento, que é gravada em um *smart card* e apresentada para a retirada do bilhete; a vantagem é que essa ficha é muito difícil de forjar e pode ser facilmente autenticada no momento de sua apresentação, servindo como comprovante real de pagamento. Essa pode se estender para comprovar o pagamento de muitos bens, tais como passagens aéreas ou pedágios;

- o uso de fichas eletrônicas para controle de acesso a sistemas, sem envolver transações financeiras: imaginemos que se queira controlar o acesso de um usuário a uma determinada rede, permitindo que ele realize apenas um número limitado de operações por dia; esse controle poderia ser feito através de micropagamentos, dando ao usuário um conjunto de fichas no início do dia e requisitando-as a cada operação subseqüente. Uma aplicação possível para um sistema desse tipo seria

o controle de acesso à Web por funcionários de empresas que queiram limitar o uso desse recurso.

Referências Bibliográficas

[1] ALBERTIN, A. L. *Comércio Eletrônico.* 2a ed. São Paulo: Atlas, 2000.

[2] ALBRECHT, C. et al. An analysis of ssl and set for electronic commerce. Disponível em: <citeseer.nj.nec.com/252522.html>.

[3] ANDERSON, R.; MANIFAVAS, C.; SUTHERLAND, C. Netcard – a practical electronic cash system. Disponível em: <citeseer.nj.nec.com/168207.html>.

[4] ECONOMIST, T. The economist: a survey of electronic commerce. The Economist, v.343, n. 8016. maio 1997. Disponível em: <http://www.economist.com/surveys/displayStory.cfm?story_id=596285>.

[5] HALLER, N. M. The S/KEY one-time password system. In: *Proceedings of the Symposium on Network and Distributed System Security.* [S.l.: s.n.], 1994. p. 151–157. Disponível em: <citeseer.nj.nec.com/haller94skey.html>.

[6] HASSLER, V. et al. Mimi: a java implementation of the micromint scheme. 1997. Disponível em: <citeseer.nj.nec.com/hassler97mimi.html>.

[7] HAUSER, R.; STEINER, M.; WAIDNER, M. *Micro-Payments based on iKP.* [S.l.], 1996. Relatório técnico. Disponível em: <citeseer.nj.nec.com/hauser96micropayment.html>.

[8] JARECKI, S.; ODLYZKO, A. An efficient micropayment system based on probabilistic polling. *Lecture Notes in Computer Science*, v. 1318, p. 173–191, 1997. Disponível em: <citeseer.nj.nec.com/jarecki97efficient.html>.

[9] MANASSE, M. et al. The millicent protocol for inexpensive electronic commerce. 1995. Disponível em: <http://www.w3.org/Conferences/WWW4/Papers/246/>.

[10] MICALI, S.; RIVEST, R. L. Micropayments revisited. In: *CT-RSA.* [S.l.: s.n.], 2002. p. 149–163. Disponível em: <citeseer.nj.nec.com/502775.html>.

[11] MU, Y.; VARADHARAJAN, V.; LIN, Y.-X. New micropayment schemes based on paywords. In: *Australasian Conference on Information Security and Privacy.* [S.l.: s.n.], 1997. p. 283–293. Disponível em: <citeseer.nj.nec.com/mu97new.html>.

[12] NIST. Fips publication 180: Secure hash standard (shs). National Institute of Standards and Technology. Maio 1993.

[13] NÉTO, J. C. *Criptografia - Uma Implementação do Protocolo de Micropagamento Payword.* Dissertação (Mestrado) — Instituto de Matemática e Estatística da Universidade de São Paulo, 2002.

[14] PEDERSEN, T. P. Electronic payments of small amounts. In: *Security Protocols Workshop.* [S.l.: s.n.], 1996. p. 59–68. Disponível em: <citeseer.nj.nec.com/pedersen96electronic.html>.

[15] POUTANEN, T.; HINTON, H.; STUMM, M. NetCents: A lightweight protocol for secure micropayments. Disponível em: <citeseer.nj.nec.com/poutanen98netcents.html>.

[16] RIVEST., R. The md5 message-digest algorithm. Request For Comments: 1321. Abril 1992.

[17] RIVEST, R. L. Electronic lottery tickets as micropayments. In: HIRSCHFELD, R. (Ed.). *Financial Cryptography.* Anguilla, British West Indies: Springer, 1997. p. 307–314. Disponível em: <citeseer.nj.nec.com/rivest98electronic.html>.

[18] RIVEST, R. L.; SHAMIR, A. Payword and micromint: Two simple micropayment schemes. In: *Security Protocols Workshop.* [S.l.: s.n.], 1996. p. 69–87. Disponível em: <citeseer.nj.nec.com/rivest96payword.html>.

[19] RIVEST, R. L.; SHAMIR, A.; ADELMAN, L. M. *A Method for Obtaining Digital Signatures and Public-key Cryptosystem.* [S.l.], 1977. 15 p. Disponível em: <citeseer.nj.nec.com/rivest78method.html>.

[20] SCHMIDT, C.; MüLLER, R. A framework for micropayment evaluation. Disponível em: <http://citeseer.nj.nec.com/172517.html>.

[21] SCHWEITZER, C. M. et al. *Sistema para Controle de Distribuição Segura de Imagens.* São Paulo, 2001. Relatório técnico LARC - PCS - EPUSP / Scopus.

[22] SET. Secure electronic transaction llc. Set Specification Books. Disponível em: <http://www.setco.org/set_specifications.html>.

[23] STERN, J.; VAUDENAY, S. SVP: A flexible micropayment scheme. In: *Financial Cryptography.* [S.l.: s.n.], 1997. p. 161–172. Disponível em: <citeseer.nj.nec.com/365000.html>.

[24] YEN, S.-M.; KU, P.-Y. *Improved Micro-payment Scheme.* [S.l.], 1998. Disponível em: <citeseer.nj.nec.com/yen98improved.html>.