Um Sistema de Padrões para Software Criptográfico Orientado a Objetos

Alexandre M. Braga *Cecília M. F. Rubira* *Ricardo Dahab*

Instituto de Computação - UNICAMP.
13081-970 Campinas-SP-Brasil. Caixa Postal 6176
fone/fax:+55+19+7885842
e-mail:{972314,cmrubira,rdahab}@dcc.unicamp.br

Sumário

Neste trabalho, um sistema de padrões [BMR+96] para software criptográfico orientado a objetos é proposto. Neste sistema, os aspectos da arquitetura de software criptográfico são abordados como microarquiteturas [GHJV93] interrelacionadas. Quatro padrões são classificados de acordo com os objetivos fundamentais da criptografia [MvOV96](ver Apêndice A). Os padrões são: *Sigilo da Informação*, *Integridade da Mensagem*, *Autenticação da Mensagem* e *Autenticação do Remetente*. Outros quatro padrões são obtidos a partir de combinações dos anteriores: *Sigilo com Autenticação*, *Sigilo com Integridade*, *Sigilo com Assinatura* e *Assinatura com Apêndice*. Os oito padrões criptográficos possuem estrutura e dinâmica comuns e, por isso, podem ser representados por um padrão mais genérico, chamado *Metapadrão Criptográfico*. Os oito padrões e o *Metapadrão Criptográfico* podem ser dispostos como vértices de um grafo direcionado acíclico no qual um percurso documenta uma sequência de decisões de projeto.

Palavras-chave: criptografia, sistema de padrões, padrões de projeto, arquitetura de software, orientação a objetos.

Abstract

In this work, a system of patterns [BMR+96] for object-oriented cryptographic software is proposed. In this system, architectural aspects of cryptographic software are treated as interrelated microarchitectures [GHJV93]. A classification based on fundamental objectives of cryptography [MvOV96](see Appendix A) is used to organize a set of patterns; they are: *Information Secrecy*, *Message Integrity*, *Message Authentication*, and *Sender Authentication*. By combination of these patterns, another set of four patterns is obtained: *Secrecy with Authentication*, *Secrecy with Signature*, *Secrecy with Integrity*, and *Signature with Appendix*. These eight cryptographic patterns have the same structural and dynamic properties and can be modeled by a generic object-oriented *Cryptographic Metapattern*. A directed acyclic graph can be constructed using the eight cryptographic patterns and the generic *Cryptographic Metapattern*. A walk on that graph documents a sequence of design decisions.

Key words: cryptography, system of patterns, design patterns, software architecture, object orientation.

1 Introdução

Tradicionalmente, a Ciência da Computação e, em particular, o desenvolvimento de software criptográfico, tem se preocupado com o projeto e implementação de algoritmos e suas estruturas de dados. Nos últimos anos, a preocupação com a organização de alto nível dos componentes de sistemas grandes fez surgir uma área nova de pesquisa, a arquitetura de software: a estrutura dos componentes de um programa/sistema, seus interrelacionamentos, princípios e diretrizes que governam seu projeto e evolução temporal [GP95]. Muito investimento tem sido feito em software criptográfico. A criptografia está sendo integrada a aplicações de vários domínios, às vezes se tornando uma característica padrão em, por exemplo, processadores de texto, sistemas de bancos de dados e planilhas eletrônicas. Ela também é um componente importante no desenvolvimento de protocolos de rede usados na construção de aplicações seguras para comércio eletrônico [BBDR98], *home banking*, bancos de dados distribuídos, correio eletrônico seguro, *intranets* privadas e teleconferências, entre outras aplicações. Aspectos da arquitetura de software criptográfico são abordados neste trabalho como microarquiteturas [GHJV93] interrelacionadas e específicas para o domínio da criptografia. A contribuição deste trabalho é o estabelecimento de um sistema de padrões para software criptográfico orientado a objetos, capaz de orientar o desenvolvimento e uso de um *framework* [Pre95] para este domímio. O texto a seguir está organizado da seguinte forma. Um sistema de padrões para software criptográfico é proposto na Seção 2. Na Seção 3, os padrões criptográficos são combinados e produzem padrões novos, estendendo o sistema proposto. Na Seção 4, um *Metapadrão Criptográfico* genérico é proposto baseado nos padrões criptográficos. Conclusões e trabalhos futuros são apresentados na Seção 5. O Apêndice A cobre conceitos básicos de criptografia.

2 Sistema de Padrões para Software Criptográfico

Grupos de padrões podem ser organizados de três maneiras, de acordo com o grau de relacionamento entre os padrões constituintes do grupo: (*i*) catálogos, (*ii*) sistemas e (*iii*) linguagens de padrões [BMR+96]. Catálogos de padrões apresentam cada padrão de forma relativamente independente; em outras palavras, catálogos de padrões são coleções de soluções relativamente independentes para problemas comuns de projeto [SFJ96]. Sistemas de padrões dão ênfase aos relacionamentos entre os padrões e cobrem aspectos específicos de um domínio. Sistemas de padrões são linguagens incompletas de padrões. Linguagens de padrões cobrem todos os aspectos importantes em um domínio de aplicação e são completas, isto é, linguagens de padrões apresentam famílias de padrões relacionados que cobrem domínios e disciplinas específicos [SFJ96]. Linguagens e sistemas de padrões apoiam reutilização de projeto e arquiteturas de software em escala maior que os padrões isolados. Neste trabalho, um sistema de padrões para software criptográfico orientado a objetos é proposto. Este sistema é definido como um conjunto de padrões criptográficos juntamente com diretrizes para sua combinação e uso prático no desenvolvimento de software criptográfico orientado a objetos. Os padrões descritos cobrem aspectos específicos da construção de software criptográfico orientado a objetos. Uma classificação e combinações dos padrões apresentados também são propostas.

2.1 Padrões de Projeto Criptográficos

Os padrões são classificados em quatro categorias de acordo com os objetivos criptográficos descritos no Apêndice A: sigilo, integridade, autenticação e não-repúdio. Um padrão de projeto criptográfico correspondente a cada categoria é apresentado no formato para descrição de padrões Contexto-Problema-Solução-Exemplo [Lea94, BMR+96]. São eles: *Sigilo da Informação*, *Integridade da Mensagem*, *Autenticação da Mensagem* e *Autenticação do Remetente*. Outros quatro padrões são obtidos a partir de combinações dos anteriores e apresentados de forma breve e informal na Seção 3: *Sigilo com Integridade*, *Sigilo com Autenticação*, *Sigilo com Assinatura* e *Assinatura com Apêndice*. A Tabela 2.1 lista os padrões criptográficos desta seção e as categorias correspondentes.

Estes padrões foram identificados durante o desenvolvimento de um *framework* de aplicação para venda e distribuição *on-line* de produtos baseados em micropagamentos [BDR98], na análise dos aspectos de segurança das aplicações para comércio eletrônico [BBDR98] e nos serviços oferecidos por algumas APIs criptográficas [Deg97, Kal95, css97]. Por isso, exemplos da utilização dos padrões criptográficos estão relacionados a estes domínios. É importante esclarecer que estas aplicações ainda não implementam os padrões criptográficos explicitamente, tão pouco foram projetadas com estes padrões em mente, uma vez que eles ainda não haviam sido documentados, mas elas usam soluções recorrentes para problemas comuns da criptografia. Tais soluções estão sendo documentadas como padrões de projeto neste artigo. A notação usada nos diagramas de classe e de troca de mensagens apresentados em seguida é semelhante aquela usada por Gamma *et al.* [GHJV94]. Nos diagramas de classe, retângulos com bordas de linhas pontilhadas contém pseudo-código para um método da classe correspondente.

Padrão	Categoria	Objetivo do Padrão
Sigilo da Informação	Sigilo	Manter a informação em sigilo
Integridade da Mensagem	Integridade	Inibir a corrupção da informação
Autenticação da Mensagem	Autenticação	Autenticar a origem da informação
Autenticação do Remetente	Não-repúdio	Inibir a negação da informação

Tabela 2.1: Padrões de Projeto Criptográficos.

2.1.1 Padrão Sigilo da Informação

Contexto: Alice deseja enviar mensagens para Bob, mas as mensagens são transmitidas de modo que um adversário possa observá-las.

Problema: Encontrar um modo pelo qual o conteúdo da mensagem enviada por Alice não possa ser determinado pelo adversário.

Solução: Este padrão proporciona apoio ao ciframento e deciframento de dados. Alice e Bob concordam previamente em usar um cifrador/decifrador criptográfico e compartilham uma chave secreta (se criptografia de chave pública for usada, Bob possui a chave pública de Alice). Bob cifra a mensagem e a envia para Alice. Alice decifra a mensagem cifrada por Bob e obtém a mensagem original.

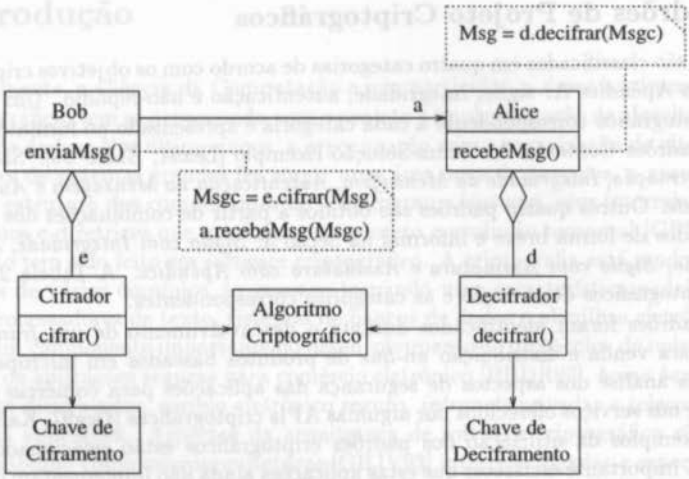

Figura 1: Sigilo de Informações.

Exemplo: Ciframento e deciframento é o serviço criptográfico básico e todas as APIs criptográficas oferecem apoio a este serviço, algumas com restrições quanto à exportação.

O adversário pode ser capaz de interceptar a mensagem cifrada, mas sem conhecer a chave de deciframento, não é possível determinar o conteúdo original da mensagem. A estrutura do padrão está na Figura 1 e a dinâmica de troca de mensagens, na Figura 2.

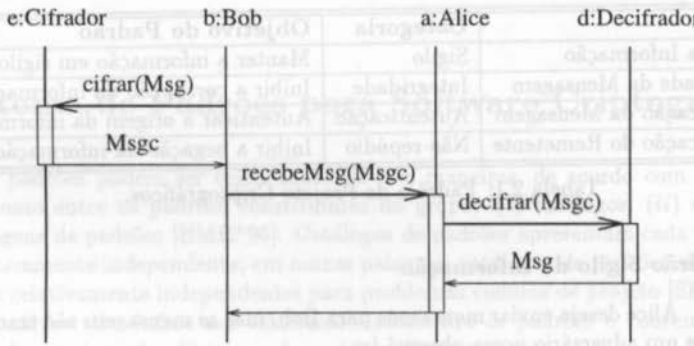

Figura 2: Dinâmica do Sigilo de Informações.

2.1.2 Padrão Integridade da Mensagem

Contexto: Alice deseja enviar uma mensagem muito grande para Bob, mas eles não compartilham chaves para ciframento e deciframento.

Problema: Determinar se a mensagem foi alterada ou substituída entre o instante de envio por Alice e o recebimento por Bob.

Solução: Alice e Bob concordam em usar um código para detecção de manipulação (MDC). Alice calcula o MDC para a mensagem e informa Bob do valor obtido, de modo íntegro. Ao receber a mensagem, Bob calcula o MDC e o compara com o valor obtido de Alice. Se os valores forem iguais, a mensagem é aceita como genuína. A estrutura deste padrão está na Figura 3.

Exemplo: MDCs podem ser usados como identificadores únicos de moedas eletrônicas em aplicações de comércio eletrônico com micropagamentos baseados em cadeias de *hash* [BBDR98]. Nestas aplicações, o módulo do pagador contém um porta-moedas eletrônico que usa uma função de *hash* para gerar cadeias; moedas são elementos dessa cadeia. O pagador envia as moedas para um objeto remoto, o recebedor, o qual a verifica e contabiliza o pagamento.

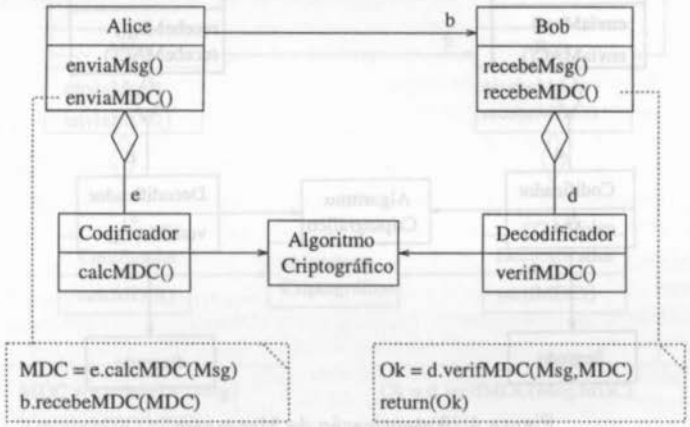

Figura 3: Integridade da Mensagem.

Usos adicionais do MDC são a detecção de modificação por vírus em arquivos e a geração de *hash* de *passphrases* para reduzi-las a um valor capaz de ser usado como chave criptográfica. *Passphrases* são como senhas, mas de comprimento maior.

2.1.3 Padrão Autenticação da Mensagem

Contexto: Alice e Bob desejam trocar mensagens entre si, mas as mensagens são transmitidas de modo que a inserção de mensagens de um adversário no canal de comunicação é inevitável.

Problema: Determinar um método pelo qual as mensagens genuínas possam ser diferenciadas daquelas enviadas pelo adversário.

Solução: Alice e Bob concordam em compartilhar uma chave secreta e um algoritmo criptográfico para geração de MACs (*Message Autentication Codes*). Alice calcula o MAC para a mensagem e envia ambos, mensagem e MAC, para Bob. Bob calcula novamente o MAC e o compara ao valor recebido. Se os valores forem iguais, a mensagem é considerada

genuína e somente Alice pode tê-la enviado, uma vez que, além de Bob, somente Alice conhece a chave secreta para calcular o MAC corretamente. A Figura 4 mostra a estrutura do padrão.

Exemplo: A CSSM-API [css97] apoia este serviço oferecendo interfaces padronizadas para cálculo e verificação de MACs.

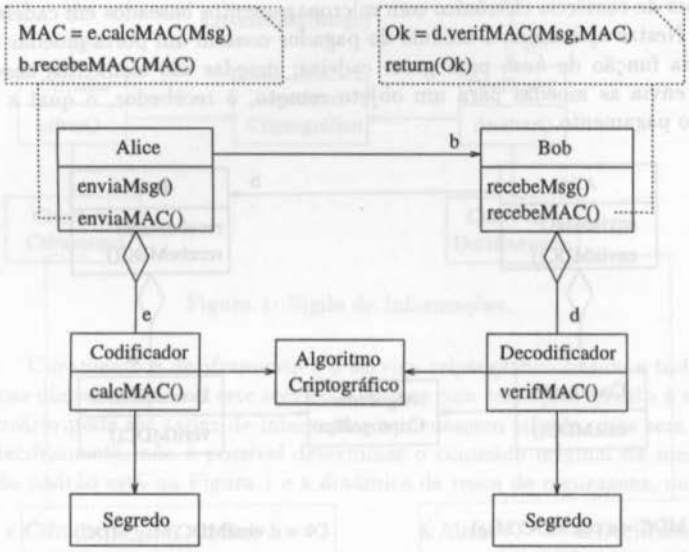

Figura 4: Autenticação de Mensagem.

Uma preocupação adicional em relação à autenticação e integridade é a atualidade da mensagem. Um adversário pode interceptar uma mensagem e enviá-la muitas vezes caso ela não possua uma indicação de atualidade. A solução é inserir na mensagem uma informação variável no tempo, por exemplo, *timestamps*, números sequenciais e *nonces* (números aleatórios únicos). Bob pode considerar a mensagem genuína de acordo com algum critério baseado na informação temporal, por exemplo, a sequência numérica deve ser sempre crescente ou o *timestamp* deve estar no interior de um intervalo de tempo determinado.

2.1.4 Padrão Autenticação do Remetente

Contexto: Alice e Bob desejam trocar mensagens entre si, mas as mensagens são transmitidas de modo que a inserção de mensagens de um adversário no canal de comunicação é inevitável. Além disso, Alice sempre pode negar ter enviado uma mensagem genuína para Bob.

Problema: Garantir que a mensagem recebida de Alice é genuína e autêntica quanto ao remetente.

Solução: Alice e Bob concordam em usar um código para detecção de manipulação (MDC). Alice calcula o MDC para a mensagem e informa Bob do valor obtido, de modo íntegro. Ao receber a mensagem, Bob calcula o MDC e o compara com o valor obtido de Alice. Se os valores forem iguais, a mensagem é aceita como genuína. A estrutura deste padrão está na Figura 3.

Exemplo: MDCs podem ser usados como identificadores únicos de moedas eletrônicas em aplicações de comércio eletrônico com micropagamentos baseados em cadeias de *hash* [BBDR98]. Nestas aplicações, o módulo do pagador contém um porta-moedas eletrônico que usa uma função de *hash* para gerar cadeias; moedas são elementos dessa cadeia. O pagador envia as moedas para um objeto remoto, o recebedor, o qual a verifica e contabiliza o pagamento.

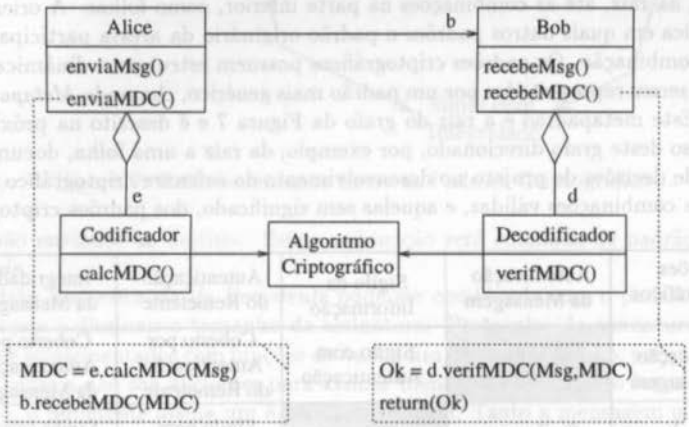

Figura 3: Integridade da Mensagem.

Usos adicionais do MDC são a detecção de modificação por vírus em arquivos e a geração de *hash* de *passphrases* para reduzi-las a um valor capaz de ser usado como chave criptográfica. *Passphrases* são como senhas, mas de comprimento maior.

2.1.3 Padrão Autenticação da Mensagem

Contexto: Alice e Bob desejam trocar mensagens entre si, mas as mensagens são transmitidas de modo que a inserção de mensagens de um adversário no canal de comunicação é inevitável.

Problema: Determinar um método pelo qual as mensagens genuínas possam ser diferenciadas daquelas enviadas pelo adversário.

Solução: Alice e Bob concordam em compartilhar uma chave secreta e um algoritmo criptográfico para geração de MACs (*Message Autentication Codes*). Alice calcula o MAC para a mensagem e envia ambos, mensagem e MAC, para Bob. Bob calcula novamente o MAC e o compara ao valor recebido. Se os valores forem iguais, a mensagem é considerada

3 Relacionamentos entre os Padrões

Padrões podem ser relacionados por integração, variação ou combinação [BMR+96]. No primeiro caso, os componentes e relacionamentos internos aos padrões são descritos por outros padrões menores. No segundo caso, um padrão e suas variações descrevem soluções para problemas semelhantes. Finalmente, padrões podem ser combinados em estruturas mais complexas e no mesmo nível de abstração. Se um problema complexo envolve aspectos não tratados por um único padrão, vários padrões são combinados e cada um resolve uma parte do problema. Os padrões criptográficos podem ser combinados para solucionar problemas de segurança nos quais mais de um dos objetivos da criptografia devem ser alcançados. As combinações também podem ser descritas como padrões. A Figura 7 apresenta os padrões criptográficos já descritos, e aqueles formados por combinações dos anteriores, organizados como vértices de um grafo direcionado acíclico, iniciando pelo padrão mais geral, na raiz, até as combinações na parte inferior, como folhas. A orientação das arestas indica em quais outros padrões o padrão originário da aresta participa como elemento de combinação. Os padrões criptográficos possuem estrutura e dinâmica comuns e capazes de serem representadas por um padrão mais genérico, chamado *Metapadrão Criptográfico*. Este metapadrão é a raiz do grafo da Figura 7 e é descrito na próxima seção. Um percurso deste grafo direcionado, por exemplo, da raiz a uma folha, documenta uma sequência de decisões de projeto no desenvolvimento do software criptográfico. A Figura 6 mostra as combinações válidas, e aquelas sem significado, dos padrões criptográficos.

Padrões Criptográficos	Autenticação da Mensagem	Sigilo da Informação	Autenticação do Remetente	Integridade da Mensagem
Autenticação da Mensagem		Sigilo com Autenticação	Coberto por Autenticação do Remetente	Coberto por Autenticação da Mensagem
Sigilo da Informação	Sigilo com Autenticação		Sigilo com Assinatura	Sigilo com Integridade
Autenticação do Remetente	Coberto por Autenticação do Remetente	Sigilo com Assinatura		Assinatura com Apêndice
Integridade da Mensagem	Coberto por Autenticação da Mensagem	Sigilo com Integridade	Assinatura com Apêndice	
	Padrões Criptográficos Combinados			

Figura 6: Padrões Criptográficos e Suas Combinações Mais Comuns.

Se tanto sigilo quanto autenticação e integridade são necessários, o padrão *Sigilo da Informação* pode ser usado em conjunto com o padrão *Autenticação da Mensagem*. O MAC deve ser calculado sobre a mensagem original. Tanto a mensagem cifrada quanto o MAC são enviados ao destino e a chave secreta de geração do MAC deve ser diferente daquela usada para ciframento e deciframento. Esta combinação será chamada de padrão *Sigilo com Autenticação*. Caso somente integridade e sigilo sejam necessários, o padrão *Sigilo da Informação* pode ser usado em conjunto com o padrão *Integridade da Mensagem*. O MDC deve ser calculado sobre a mensagem original. Tanto a mensagem cifrada quanto

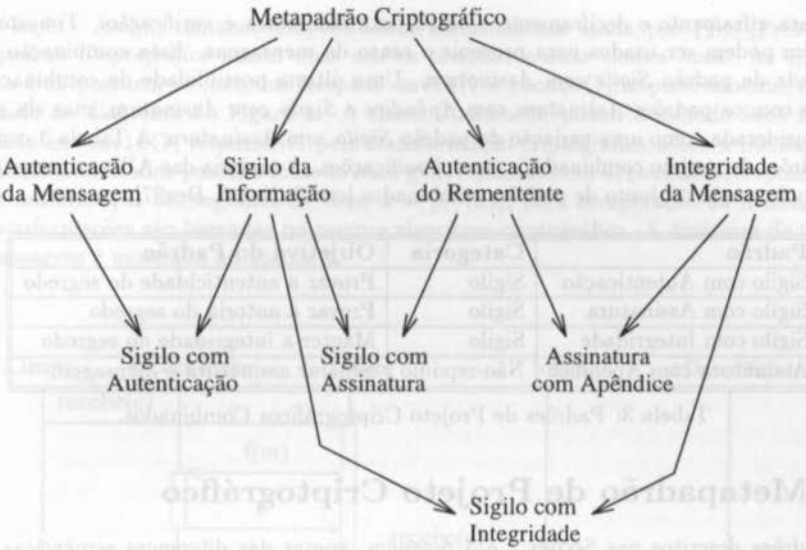

Figura 7: Relacionamentos Entre os Padrões Criptográficos.

o MDC são enviados ao destino. Esta combinação será chamada de padrão *Sigilo com Integridade*.

O padrão *Autenticação do Remetente* pode ser combinado com o padrão *Integridade da Mensagem* e diminuir o tamanho da assinatura. Protocolos de assinatura digital são geralmente implementados com funções de *hash* a fim de poupar tempo, porque algoritmos de chave pública não são eficientes para assinar mensagens longas. No lugar de assinar a mensagem, o remetente assina um *hash* da mensagem. Tanto a mensagem quanto o *hash* assinado são enviados ao destino. O destinatário decifra o *hash* assinado e o compara àquele calculado a partir da mensagem recebida; se forem iguais, a assinatura é válida. Alguns benefícios deste esquema combinado são desempenho maior, requisitos de memória menores e possibilidade de manter mensagem e assinatura separadas. Por outro lado, um acordo prévio deve ser feito sobre quais algoritmos de assinatura e *hash* serão usados. Esta combinação será chamada de padrão *Assinatura com Apêndice*.

Os padrões *Autenticação do Remetente* e *Sigilo da Informação* podem ser combinados para proporcionar prova de autoria e privacidade. Os protocolos de assinatura digital e ciframento com chave pública são combinados da seguinte forma [Sch96]:

1. Alice assina a mensagem com sua chave privada.
2. Alice cifra, com a chave pública de Bob, a mensagem assinada e a envia para Bob.
3. Bob decifra a mensagem com sua chave privada.
4. Bob verifica, com a chave pública de Alice, a mensagem assinada e recupera a mensagem original.

Com este protocolo combinado, um adversário não pode remover a assinatura de uma mensagem cifrada e adicioná-la a outra mensagem. Alice deve possuir dois pares de chaves,

um para ciframento e deciframento e outro para assinatura e verificação. *Timestamps* também podem ser usados para prevenir o reuso de mensagens. Esta combinação será chamada de padrão *Sigilo com Assinatura*. Uma última possiblidade de combinação é obtida com os padrões *Assinatura com Apêndice* e *Sigilo com Assinatura*, mas ela pode ser considerada como uma variação do padrão *Sigilo com Assinatura*. A Tabela 3 contém os padrões de projeto combinados e suas classificações. A maioria das APIs criptográficas dão apoio a este conjunto de padrões combinados [css97, Kal95, Deg97].

Padrão	Categoria	Objetivo do Padrão
Sigilo com Autenticação	Sigilo	Provar a autenticidade do segredo
Sigilo com Assinatura	Sigilo	Provar a autoria do segredo
Sigilo com Integridade	Sigilo	Manter a integridade do segredo
Assinatura com Apêndice	Não-repúdio	Separar assinatura e mensagem

Tabela 3: Padrões de Projeto Criptográficos Combinados.

4 Metapadrão de Projeto Criptográfico

Os padrões descritos nas Seções 2 e 3 possuem, apesar das diferenças semânticas, essencialmente as mesmas estrutura e dinâmica de troca de mensagens. Estas estrutura e dinâmica comuns podem ser representadas como um padrão de projeto criptográfico genérico, localizado um nível de abstração acima daqueles padrões criptográficos anteriores. Além disso, todos aqueles padrões podem ser descritos nos moldes deste padrão mais geral. Este novo padrão, pelos motivos acima, é chamado de *Metapadrão Criptográfico*. O metapadrão estabelece antecipadamente a microarquitetura para as implementações dos padrões criptográficos.

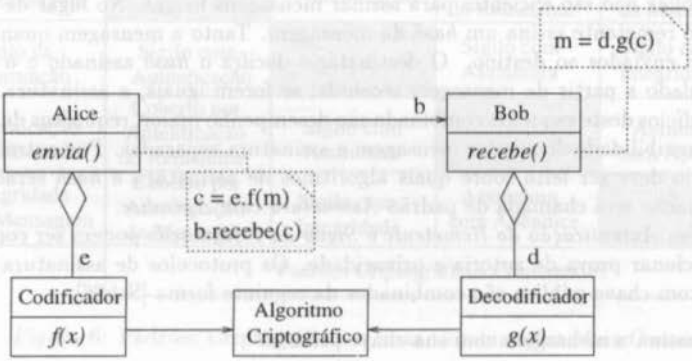

Figura 8: Metapadrão Criptográfico.

O *Metapadrão Criptográfico* não deve ser confundido com os metapadrões descritos por Pree [Pre95] porque este metapadrão somente representa uma abstração de nível mais alto em relação aos padrões da seção anterior. Isto é, o *Metapadrão Criptográfico* é um padrão sobre padrões criptográficos. Por outro lado, os metapadrões de Pree possuem propósito mais geral e fazem parte de qualquer padrão de projeto orientado a objeto, incluindo os padrões de projeto criptográficos da seção 2 e o *Metapadrão Criptográfico*

desta seção. Assim, tomando emprestada a nomenclatura usada por Pree [Pre95], o *Metapadrão Criptográfico* possui duas classes *template* e duas classes *hook*. As classes Alice e Bob possuem os métodos *template* `envia()` e `recebe()`, respectivamente, como mostrado no diagrama da Figura 8. A classe Codificador possui o método *hook* `f(x)`, chamado em `envia()`, responsável pela transformação criptográfica sobre a mensagem. A classe Decodificador possui o método *hook* `g(x)`, chamado em `b.recebe()`, responsável pela transformação criptográfica inversa, $x = g(f(x))$, para recuperação da mensagem. As transformações são baseadas no mesmo algoritmo criptográfico. A dinâmica de troca de mensagens é mostrada na Figura 9.

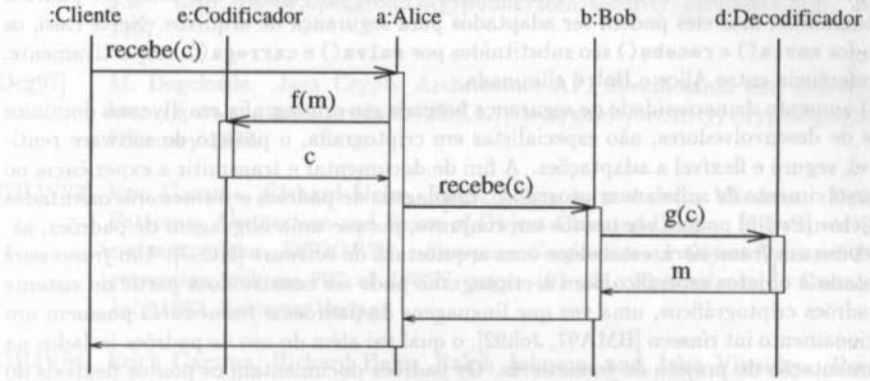

Figura 9: Dinâmica do Metapadrão Criptográfico.

A microarquitetura de software criptográfico proposta com o *Metapadrão Criptográfico* pode ser combinada a outros padrões de projeto. O padrão *Strategy* [GHJV94, 315] pode ser usado para generalizar o algoritmo criptográfico. O padrão *Bridge* [GHJV94, 151] pode ser usado na separação entre interface do algoritmo e implementações específicas. O padrão *Abstract Factory* [GHJV94, 87] pode ser usado na etapa prévia de negociação entre Alice e Bob sobre qual algoritmo e implementação usar. Os padrões *Observer* [GHJV94, 293], *Proxy* [GHJV94, 207] e *Client-Dispatcher-Server* [BMR+96, 323] podem ser usados para tornar transparente a comunicação entre Alice e Bob, proporcionando transparência de localização. O padrão *Forwarder-Receiver* [BMR+96, 307] pode ser combinado aos padrões criptográficos para proporcionar comunicação transparente entre processos. Neste caso, Alice seria parte do *Forwarder* e Bob, parte do *Receiver*. O padrão *State* [GHJV94, 305] também pode ser usado na implementação de comportamento dependente do estado. Como, por exemplo, ligar e desligar a segurança do canal de comunicação. Padrões para aspectos de segurança de mais alto nível das aplicações são propostos por Yoder e Barcalow [YB97], mas não tratam de questões de segurança de mais baixo nível como, por exemplo, criptografia. De fato, um padrão chamado *Security Access Layer* [YB97, 6] é proposto a fim de separar os serviços de segurança e criptográficos de baixo nível do restante da aplicação, resultando em uma arquitetura de software em camadas.

5 Conclusões e Trabalhos Futuros

Este artigo abordou aspectos de arquitetura do desenvolvimento de software criptográfico, uma área na qual, tradicionalmente, a ênfase esteve no projeto de algoritmos. Um sistema de padrões criptográficos foi proposto e cobre os seguintes aspectos da criptografia: sigilo, integridade, autenticação e não-repúdio. O sistema de padrões não cobre todos os aspectos da arquitetura de software criptográfico. Serviços criptográficos auxiliares como a geração de números aleatórios e acordo, geração e distribuição de chaves, assim como outras variações daqueles padrões tratados aqui, não foram abordados e podem originar outros padrões. Por outro lado, serviços que não possam ser documentados como instâncias do *Metapadrão Criptográfico* não serão considerados padrões pertencentes ao sistema de padrões deste artigo. O uso de comunicação *on-line* foi suposto em todos os padrões criptográficos, mas eles podem ser adaptados para segurança de arquivos. Neste caso, os métodos `envia()` e `recebe()` são substituídos por `salva()` e `carrega()`, respectivamente, e a referência entre Alice e Bob é eliminada.

O aumento da necessidade de segurança baseada em criptografia em diversos domínios exige de desenvolvedores, não especialistas em criptografia, o projeto de software reutilizável, seguro e flexível a adaptações. A fim de documentar e transmitir a experiência no desenvolvimento de software criptográfico, linguagens de padrões e *frameworks* orientados a objetos [Pre95] podem ser usados em conjunto, porque uma linguagem de padrões, assim como um *framework*, estabelece uma arquitetura de software [KC97]. Um *framework* orientado a objetos específico para a criptografia pode ser construído a partir do sistema de padrões criptográficos, uma vez que linguagens de padrões e *frameworks* possuem um relacionamento int rínseco [BMA97, Joh92], o qual vai além do uso de padrões isolados na documentação do projeto de *frameworks*. Os padrões documentam os pontos flexíveis do *framework* e o grafo direcionado acíclico das dependências entre os padrões documentam e orientam a sequência de decisões de projeto a respeito de como usar o *framework*. Um *framework* criptográfico baseado nos padrões propostos neste trabalho está sendo projetado e um protótipo está sendo implementado na linguagem Java.

6 Agradecimentos

Este trabalho obteve apoio parcial do CNPq, FAPESP (processos número 97/11128-3 para Alexandre Melo Braga e 96/15329 para LSD-IC-UNICAMP) e ProNEx (Complexidade de Estruturas Discretas, número 107/97, para Ricardo Dahab). Gostariamos de agradecer aos avaliadores pelas contribuições à versão final deste artigo.

Referências

[BBDR98] Alexandre M. Braga, Delano M. Beder, Ricardo Dahab, and Cecília M. F. Rubira. Segurança em Sistemas de Micropagamentos Eletrônicos. In Mário M. Leboute, editor, *WTF'98 - I Workshop de Tolerância a Falhas*, pages 85–90, Universidade Federal do Rio Grande do Sul - Instituto de Informática, Porto Alegre, RS, Brasil, May 1998.

[BDR98] Alexandre M. Braga, Ricardo Dahab, and Cecília M. F. Rubira. PayPerClick: Um Framework para Venda e Distribuição On-line de Publicações Baseado

em Micropagamentos. In *SBRC'98 - 16o Simpósio Brasileiro de Redes de Computadores*, page 767, Rio de Janeiro, RJ, Brasil, May 1998. Resumo Estendido.

[BMA97] Davide Brugali, Giuseppe Menga, and Amund Aarten. The Framework Life Span. *Communications of the ACM*, 40(10):65–68, October 1997.

[BMR+96] Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, and Michael Stal. *Pattern-Oriented Software Architecture: A System of Patterns*. John Wiley and Sons Ltd., Chichester, UK, 1996.

[css97] Common Security Services Manager Application Programming Interface, draft 2.0. http://www.opengroup.org/public/tech/security/pki/index.htm, June 1997.

[Deg97] M. Degeforde. Java Crypto Architecture API Specification and Reference. http://java.sun.com/products/JDK1.1/docs/guide/security/CryptoSpec.html, February 1997.

[GHJV93] Eric Gamma, Richard Helm, Ralph Johnson, and John Vlissides. Design Patterns: Abstraction and Reuse of Object-Oriented Design. In Oscar M. Nierstrasz, editor, *ECOOP'93 - European Conference on Object-Oriented Programming*, volume 707 of *LNCS*, pages 407–431, Kaiserslautern, Germany, July 1993. Springer-Verlag.

[GHJV94] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison Wesley Publishing Company, April 1994.

[GP95] David Garlan and Dewayne E. Perry. Introduction to the Special Issue on Software Architecture. *IEEE Transactions on Software Engineering*, 21(4):269–274, April 1995.

[Joh92] Ralph E. Johnson. Documenting Frameworks using Patterns. In Andreas Paepcke, editor, *OOPSLA'92 - Conference on Object-Oriented Programming Systems, Languages, and Applications*, volume 27 of *ACM SIGPLAN Notices*, pages 10–20, Vancouver, Britsh Columbia, Canada, October 1992.

[Kal95] B. Kaliski. Cryptoki: A Cryptographic Token Interface, version 1.0. http://www.rsa.com/rsalabs/pubs/PKCS/html/pkcs-11.html, April 1995.

[KC97] Norman L. Kerth and Ward Cunningham. Using Patterns to Improve our Architectural Vision. *IEEE Software*, pages 53–59, January 1997.

[Lea94] Doug Lea. Christopher Alexander: An Introduction for Object-Oriented Designers. *ACM SIGSOFT, Software Engineering Notes*, 19(1):39–46, January 1994.

[MvOV96] Alfred J. Menezes, Paul C. van Orschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[Pre95] Wolfgang Pree. *Design Patterns for Object-Oriented Software Development.* Addison-Wesley, 1995.

[Sch96] Bruce Schneier. *Applied Cryptography — Protocols, Algorithms , and Source Code in C.* John Wiley and Sons, 2nd edition, 1996.

[SFJ96] Douglas C. Schmidt, Mohamed Fayad, and Ralph E. Johnson. Software Patterns. *Communications of the ACM*, 39(10):36–39, October 1996.

[YB97] Joseph Yoder and Jeffrey Barcalow. Application Security. *PLoP'97 Conference, Washington University Technical Report 97-34*, 1997.

A Conceitos Básicos de Criptografia

Criptografia é o estudo das técnicas matemáticas relacionadas aos aspectos de segurança de informações, tais como confidencialidade, integridade de dados e não-repúdio [MvOV96]. Neste contexto, os quatro objetivos fundamentais da criptografia são [MvOV96]:

1. **Confidencialidade de informação**: manter em sigilo o conteúdo da informação, exceto daqueles autorizados a conhecê-lo.
2. **Integridade de dados**: garantia de que a informação não seja modificada sem autorização e a capacidade de detectar manipulação não autorizada.
3. **Autenticação de dados**. Existe de dois modos:
   - Autenticação de entidade: duas partes devem se identificar no início de uma comunicação.
   - Autenticação de informação: informação enviada através de um canal inseguro de comunicação deve ser autenticada quanto à origem. A autenticação de origem da informação proporciona integridade implicitamente.
4. **Não-repúdio**: serviços capazes de prevenir que uma entidade negue seus compromissos ou a realização de ações; dois exemplos são a autenticação de remetente e a autenticação de destinatário.

Criptografia de chave secreta ou de chave compartilhada é o conjunto de técnicas criptográficas caracterizadas pelo uso de algoritmos nos quais uma única chave é usada tanto para ciframento quanto para deciframento. Esta chave é um segredo entre as duas partes que trocam informações criptografadas. Criptografia de chave pública são as técnicas criptográficas caracterizadas pelo uso de algoritmos nos quais um par de chaves diferentes é usado, uma chave para ciframento e outra para deciframento. A chave de ciframento é conhecida publicamente e, por isso, é chamada chave pública. Somente aquele com acesso à chave de deciframento pode decifrar dados cifrados com a chave pública correspondente. A chave de deciframento é mantida em segredo pelo seu dono e, por isso, chama-se chave privada.

A Figura 10 representa um sistema criptográfico típico no qual as operações de ciframento e deciframento são utilizadas para proporcionar sigilo de informações. É comum chamar as partes que interagem em um sistema criptográfico de Alice e Bob. Alice realiza uma operação de ciframento, $f(m, k1)$, sobre a mensagem m e com a chave $k1$, a

Figura 10: Sistema Criptográfico Típico.

chave de ciframento. A mensagem cifrada c é enviada através de um canal inseguro de comunicação até Bob que a decifra com a chave $k2$, a chave de deciframento. A operação de deciframento, $g(c, k2)$, deve ser inversa à de ciframento, isto é, $m = g(f(m, k1), k2)$. Um sistema criptográfico deve permanecer seguro mesmo quando os algoritmos para ciframento e deciframento são conhecidos. Por isso, a segurança do sistema está no segredo da chave. As chaves são usadas aos pares, uma para cifrar o texto e outra para decifrar o texto cifrado com a chave de ciframento correspondente. Deve haver um número grande de chaves a fim de impedir que o adversário decifre o texto simplesmente aplicando cada chave possível de deciframento sobre o texto cifrado. Além dos serviços típicos de ciframento e deciframento, a criptografia oferece também outros serviços básicos como as funções de *hash* unidirecionais e assinaturas digitais.

Função de *hash* é aquela que toma como entrada uma cadeia de comprimento variável e devolve uma cadeia de comprimento fixo, geralmente menor que aquela de entrada. Funções de *hash* unidirecionais são aquelas funções de *hash* nas quais é fácil computar o *hash* da cadeia de entrada, mas é computacionalmente difícil determinar uma cadeia de entrada correspondente a um *hash* conhecido. Uma função de *hash* boa para criptografia também precisa ser resistente a colisões, isto é, deve ser difícil gerar duas cadeias de entrada que resultem no mesmo *hash*. Estas funções podem ser públicas porque não existe segredo no seu algoritmo e a segurança está na sua unidirecionalidade. Funções de *hash* também são chamadas de códigos de detecção de manipulação (MDC - *Manipulation Detection Code*) e garantem integridade de dados. MAC (*Message Autentication Code*) é o valor produzido por uma função de *hash* com chave. Ele tem todas as propriedades de uma boa função de *hash* criptográfica e inclui uma chave. O MAC somente pode ser verificado por alguém com conhecimento da chave e proporciona autenticação sem sigilo e integridade implícita.

Assinaturas digitais são equivalentes eletrônicos da assinatura em papel. Um protocolo básico para assinaturas digitais baseado em criptografia de chave pública é o seguinte [Sch96]:

1. Alice cifra a mensagem com sua chave privada, assinando-a.
2. Alice envia a mensagem assinada para Bob.
3. Bob decifra a mensagem com a chave pública de Alice, verificando-a.

Assinaturas digitais devem obedecer a um conjunto de características [Sch96]: (*i*) devem ser autênticas. Quando Bob verifica uma mensagem com a chave pública de Alice, ele sabe que Alice a assinou. (*ii*) Devem ser de falsificação difícil. Apenas Alice conhece sua chave privada. (*iii*) Não são reutilizáveis. A assinatura é uma função da mensagem e não pode ser usada em mensagens diferentes. (*iv*) A mensagem assinada é inalterável. Se houver qualquer modificação na mensagem, a assinatura não poderá mais ser verificada com a chave pública de Alice. (*v*) Não podem ser negadas. Bob não precisa da ajuda de Alice para verificar a assinatura.