UMA PROPOSTA DE MODELO SEGURO PARA GESTÃO DE IDENTIDADE EM SISTEMAS MÓVEIS DE PAGAMENTOS ELETRÔNICOS

Marco Antônio Coutinho

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientador: Luís Felipe Magalhães de Moraes

Rio de Janeiro
Março de 2015

UMA PROPOSTA DE MODELO SEGURO PARA GESTÃO DE IDENTIDADE EM SISTEMAS MÓVEIS DE PAGAMENTOS ELETRÔNICOS

Marco Antônio Coutinho

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

Prof. Luís Felipe Magalhães de Moraes, Ph.D.

Prof. Claudio Luís de Amorim, Ph.D.

Prof. Marcio Portes de Albuquerque, Ph.D.

RIO DE JANEIRO, RJ – BRASIL
MARÇO DE 2015

Coutinho, Marco Antônio

uma proposta de modelo seguro para gestão de identidade em sistemas móveis de pagamentos eletrônicos/Marco Antônio Coutinho. – Rio de Janeiro: UFRJ/COPPE, 2015.

XV, 95 p.: il.; 29, 7cm.

Orientador: Luís Felipe Magalhães de Moraes

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2015.

Referências Bibliográficas: p. 79 – 84.

1. Segurança da Informação. 2. Pagamentos Móveis. 3. Android. I. Moraes, Luís Felipe Magalhães de. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

À minha esposa Larisse Santana,
por sua lealdade, amor,
companheirismo e, por vezes,
compaixão.

Agradecimentos

Agradeço a minha mãe Laurita Maria, minhas irmãs Fátima e Lígia Coutinho pelos ensinamentos eternos sobre amizade, telerância e o valor do trabalho. Agradeço a minha esposa, Larisse Teresa Santana, pelo eterno bom humor, otimismo e paciência em tantos momentos de alegria e alguns poucos de tristeza. Agradeço a minha filha, Vitória, por ter surgido em minha vida mudando todos meus conceitos sobre o amor. Agradeço ao meu orientador, Professor Luís Felipe M. de Moraes, pelos valiosos ensinamentos não apenas sobre a ciência, mas sobre diversos aspectos da vida. A todos meus professores que contribuíram nesta minha jornada, em especial aos professores da COPPE. Agradeço a todos meus companheiros de mestrado, Evandro Macedo, José Barbosa, Renato Silva e Vander Proença pelos inestimáveis momentos em que compartilhamos problemas, soluções e descontração. Por fim, a todos que participaram direta ou indiretamente desta minha vitória pessoal e pequena contribuição para a sociedade brasileira.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

UMA PROPOSTA DE MODELO SEGURO PARA GESTÃO DE IDENTIDADE EM SISTEMAS MÓVEIS DE PAGAMENTOS ELETRÔNICOS

Marco Antônio Coutinho

Março/2015

Orientador: Luís Felipe Magalhães de Moraes

Programa: Engenharia de Sistemas e Computação

A migração da vida privada para sistemas móveis, notoriamente o aparelho celular, é um fenômeno recente nas sociedades do mundo inteiro. A escalada de integração eletrônica, associada a inovações tecnológicas incrivelmente rápidas fizeram do aparelho móvel celular mais do que apenas um instrumento de comunicação, mas uma plataforma pessoal capaz de concentrar toda sorte de informações e dados críticos. As novas tecnologias destinadas a realizarem pagamentos móveis apenas são consequências desta tendência. Realizar transações financeiras pelo aparelho de comunicação móvel (um conceito bem mais abrangente e adequado para o "simples"celular) é tão óbvio, seguindo esta filosofia de integração e portabilidade, quanto enxergá-lo como um elemento para receber e realizar chamadas de voz. Esta dissertação apresenta uma contribuição em duas disciplinas correlacionadas e pertinentes para o cenário acima: segurança da comunicação entre plataforma móvel e um provedor de serviços financeiros e melhor proteção da identidade do usuário, propondo evitar que os incidentes nos atuais sistemas de pagamento ocorram em ambientes móveis.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

A SAFE MODEL FOR IDENTITY MANAGEMENT IN MOBILE ELECTRONIC PAYMENT SYSTEMS

Marco Antônio Coutinho

March/2015

Advisor: Luís Felipe Magalhães de Moraes

Department: Systems Engineering and Computer Science

The migration of privacy into mobile systems, notably the cell phone, is a recent phenomenon in societies around the world. The electronic integration climbing, combined with incredibly rapid technological innovations have made the cellular mobile device more than just a communication tool, but a personal platform able to store all sorts of information and critical data. The new applications designed for mobile payments are only consequences of this trend. Making financial transaction by using the mobile communication platform (a much more comprehensive and appropriate concept than simply cell phone) is so obvious, following this philosophy of integration and portability, as see it as an element for making and receiving voice calls. This dissertation presents a contribution in two correlated and relevant disciplines to the above scenario: security of communication between mobile platform and a financial service provider and better protection of user identity, in order to avoid common incidents in the current payment systems, occurring in a mobile environment.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

3GPP	3rd Generation Partnership Project, p. 11
AES	Advanced Encryption Standard, p. 62
ANSI	American National Standards Institute, p. 85
API	Application Programming Interface, p. 60
CTR	CounTer Mode - mode de atuação do algoritmo AES, p. 62
EMV	Europay, Mastercard e Visa, p. 9
GSM	Groupe Special Mobile, p. 11
HCE	Host Card Emulation, p. 16
ICMP	Internet Control Message Protocol, p. 6
IEC	International Electrotechnical Commission, p. 15
IETF	Internet Engineering Task Force, p. 43
IP	Internet Protocol, p. 4
ISO	International Organization for Standardization, p. 15
ITU-T	*International Telecommunication Union* Órgão internacional responsável pela padronização tecnológica nas comunicações., p. 48
M-Payment	Mobile Payment, p. 10
MS	Mobile Station, p. 9
NFC CWI	NFC Wired Interface, p. 16
NFC	Near Field Communication, p. 4
PAN	Primary Account Number, p. 58

PIN	Personal Identification Number, p. 3
POS	Point of Sale, p. 3
PRNG	Pseudo Random Number Generator, p. 5
RFC	Rquest for Comments, p. 43
RFID	Radio Frequency Identification Device, p. 13
SE	Secure Element, p. 15
SMS	Short Message Service, p. 11
TLS	*Transport Layer Security* - protocolo de comunicação criptográfica, que utiliza chave assimétrica e certificados digitais., p. 48
URA	Unidade de Resposta Audível, p. 12
USSD	Unstructured Supplementary Service Data, p. 12
WAP	Wireless Application Protocol, p. 13
i.i.d.	Independente e Identicamente Distribuído, p. 31

Capítulo 1

Introdução

Os métodos de pagamentos eletrônicos estão evoluindo para atender a uma necessidade imposta pelas sociedades sociotécnicas (LATOUR *et al.* [10]) em sintonia com os anseios do público: velocidade, portabilidade e segurança. Ainda que estes conceitos possam estar sujeitos a constantes questionamentos e pressões resultantes da evolução tecnológica e seu impacto sobre novos hábitos (cita-se o caso da segurança, diante do aumento considerável da exposição individual nas redes sociais, por exemplo), eles tendem a um lugar comum na avaliação dos usuários quando o assunto envolve transações financeiras. Neste aspecto, a utilização de redes móveis apresenta-se como uma opção de evolução natural para todos os métodos conhecidos.

Neste caso, faz-se necessário analisar esta evolução de técnicas de *mobile payment* tendo em vista os recursos disponíveis no principal sistema móvel disponível e com mais abrangência no mercado atualmente. O Android®[1] (ANDROID [11]) está presente em 80%, de acordo com a figura 1.1 extraída de IDC [1] dos aparelhos em uso. É significativo analisar os principais requisitos para uma adequada implementação de pagamento eletrônico tal como é proposto neste trabalho.

Como será visto detalhadamente em seções futuras, a segurança é um requisito central para a adoção, pelos usuários, de novos métodos envolvendo pagamentos. Parte disso, deve-se a noção óbvia de necessidades de confidencialidade e integridade quando movimentações financeiras se fazem necessárias, parte devido ao fato de que o sistema financeiro há muito solucionou o problema de transporte de dinheiro sem a necessidade de portabilidade em espécie: o sistema de cartão de crédito vem se desenvolvendo durante 6 décadas, conquistando a confiança dos usuários ávidos pelo consumo de massa e também do sistema bancário em geral. Para tanto, foi necessário investir em um padrão rigoroso visando a uniformidade de técnicas, nomenclaturas e gestão da segurança (EMVCO [7]). Graças a este esforço técnico

[1] Android tem registro de marca por Google Inc.

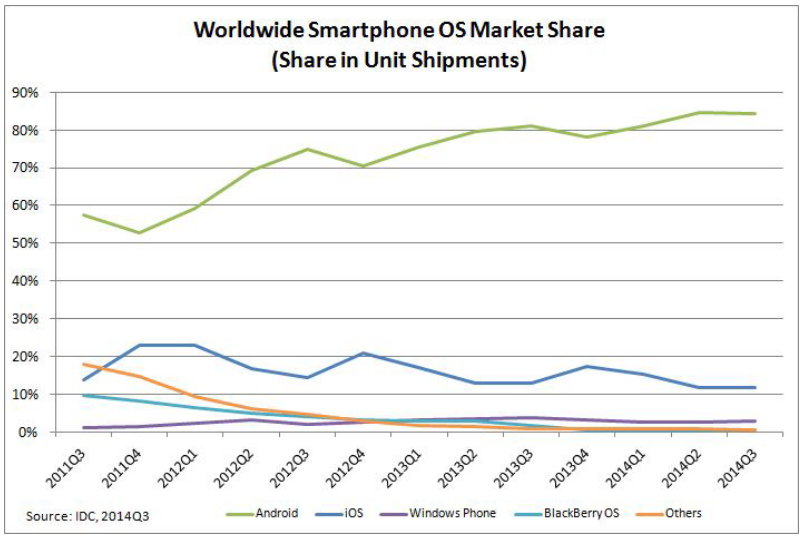


Figura 1.1: Distribuição de plataformas "smarthphone"no munddo. Fonte: IDC [1].

e político para a construção deste padrão mundial de compras eletrônicas, ainda hoje o sistema eletrônico de crédito bancário (em sua materialização mais popular através do cartão de crédito) possui altos índices de disponibilidade e confiabilidade junto aos usuários. Ao longo de sua história, este padrão universal vem se mostrando capaz de atender aos anseios dos usuários por facilidade de uso e segurança nas transações, a despeito dos frequentes casos noticiados pela mídia envolvendo fraudes e quadrilhas especializadas em roubos de identidade dos usuários.

1.1 Motivação e Definição do Problema

É esperada que a evolução tecnológica de um processo bem aceito e consolidado "sempre"seja inciada por: a) questões financeiras, em grande maioria corte/redução de custos, ou b) uma dramática mudança cultural e, por consequência, comportamental das sociedades sociotécnicas modernas. O tema desta dissertação está, sob diversos aspectos, em sintonia com ambas afirmações anteriores pois de fato a motivação das novas técnicas de pagamentos eletrônicos têm impacto direto sobre a infraestrutura necessária para a captura e processamento das transações financeiras mas, em grande parte deste esforço de evolução, encontra-se o desejo de ir ao encontro dos apelos do público usuário para simplificar o uso da tecnologia, bem como aumentar mais a portabilidade da mesma dentro do já bem sucedido computador pessoal multi proposta que se tornou o aparelho celular - em sua mais avançada versão, o *"smartphone"*. A seguir, citam-se de forma mais pragmática as principais motivações entendidas para a migração do processo de pagamento eletrônico do tradicional cartão de plástico para uma versão virtual do mesmo

dentro do aparelho celular do usuário.

- Portabilidade: A facilidade de carregar informações pessoais e outros conteúdos digitalizadas no próprio aparelho celular, tem sido um critério de escolha dos usuários de comunicação móvel para a aquisição de novas soluções tecnológicas. Uma indicação clara e definitiva desta tendência é que o próprio dinheiro já está disponível em sua versão eletrônica, o conhecido e polêmico *Bitcoin* (BITCOIN [12]). Existe grande conveniência sob o ponto de vista do usuário para não mais portar dinheiro em espécie e documentos pessoais de natureza mais crítica. Posto isso, qualquer solução de pagamento que se proponha a virtualizar tanto o crédito (desde que reconhecido devidamente pelas instituições financeiras) quanto a identidade deste usuário, estará em sintonia com esta tendência de evolução comportamental.

- Evolução Tecnológica como resposta a novos incidentes: Notícias recentes na imprensa chamaram a atenção para a necessidade de evolução dos métodos de pagamento. A despeito da eficiência e sucesso (e grande aceitação do público) dos métodos atuais, baseados em cartões com tarja magnética e/ou *SmartCard* protegido por PIN (*Personal Identification Number*), sempre provocam inseguranças notícias sobre vazamento de informações sigilosas utilizadas para transações financeiras, tais como recentemente associadas à rede varejista TARGET (ZDNET [13]) e roubo de identidades realizados por quadrilhas especializadas em tecnologias de POS (*Point Of Sale*, o aparelho para leitura de cartões em pontos de venda), as quais possuem conhecimento tecnológico para violar e roubar informações de crédito a partir destas máquinas (G1 [14]). Apesar de pouco frequentes, notícias como estas citadas sempre provocam pertubações uma vez que sugerem fragilidade no sistema Financeiro, tão crítico para o funcionamento da sociedade.

- Adesão aos serviços móveis: As operadoras de serviços móveis conseguiram grande penetração em todas as classes sociais desde o processo de privatização do setor. Apesar das constantes críticas quanto a disponibilidade e qualidade do serviço prestado, é fato que a comunicação móvel tornou-se essencial, sem o qual diversas atividades críticas não se desenvolvem. A percepção pelos usuários de que as redes móveis são meios confiáveis para a transmissão de dados críticos, tais como consultas e movimentações bancárias realizadas a partir de aplicativos nas plataformas móveis (Android e iOS, majoritariamente), indica que este meio pode ser utilizado com grande sucesso visando uma migração definitiva dos meios de pagamento.

- Pressão da Indústria para maior eficiência de custos: Existem alguns fatores que pressionam os custos para serviços de pagamentos. O primeiro é a emissão de cartões e a logística de entrega para os usuários (considerando perdas, roubos, etc.). Estes custos representam parte significativa dos custos totais envolvidos neste mercado. Logo, um esforço para a virtualização do cartão de crédito é essencial para a meta de redução dos custos de manutenção da base de usuário. Outro ponto que afeta diretamente o custo operacional das transações de pagamentos, é a rede de captura necessária para o transporte entre o ponto de venda e os sistemas de processamento da operadora de crédito. Estas redes são complexas e, em geral, terceirizadas dada a necessidade de maior gerenciamento e foco tecnológico. São aplicadas diversas tecnologias (redes IP , *Frame Relay*, dentre outras tecnologias legadas), dependendo da disponibilidade de rede das operadoras. Visando reduzir este custo de captura, as entidades financeiras buscam substituir estas redes legadas pelas redes móveis, realizando um esforço de simplificação da rede de captura e repassando paras as operadoras a responsabilidade pelo planejamento e manutenção desta etapa do processo.

- Crescimento e Popularização das tecnologias *"contactless"*: A presença de NFC (*Near Field Communication*) em grande parte dos aparelhos *"smartphones"* oferece uma grande possibilidade de popularização desta modalidade de comunicação. Inicialmente em aparelhos com sistema Android, o NFC ganhou grande impulso após a adesão da Apple[2] que iniciou a comercialização de aparelhos com NFC a partir de Setembro de 2014, coincidindo com o lançamento do seu próprio serviço de pagamentos: ApplePay®. O NFC representa a extrapolação do uso do aparelho móvel celular em processos diversos, não apenas pagamentos: segurança física (substituindo uma credencial), liberação de acesso, autenticação presencial, e muitas outras.

1.2 Objetivos do trabalho

Tal como citado anteriormente, uma vez que o Android está presente na maioria dos aparelhos em uso atualmente, é estratégico uma avaliação mais criteriosa de sua capacidade de prover segurança para serviços de pagamentos móveis. Além disso, avaliar seu potencial para lançamento de uma nova modalidade que ofereça maior segurança e comodidade para os usuários. Desta forma, o objetivo principal deste trabalho é realizar um estudo consistente do Android, do ponto de vista de segurança e apresentar uma solução de pagamentos móveis consistente com todos os apelos comerciais e culturais citados acima, observando os seguintes passos e metas:

[2]Apple Inc. www.apple.com

1. Analisar a capacidade criptográfica a partir do gerador de números pseudo-aleatórios (PRNG[3]) .

2. Propor uma abordagem que aumente a segurança criptográfica do PRNG disponível no Android (aumento de sua Entropia), visando transações eletrônicas.

3. Apresentar um modelo que tire proveito no aumento da Entropia do PRNG e diminua o risco de interceptação de informações críticas durante as transações financeiras.

1.3 Contribuições

Através da elaboração deste trabalho, as seguintes contribuições serão alcançadas:

1. Um método consistente e aceito pelo mercado para avaliação de segurança em sistemas móveis baseados em Android.

2. Uma proposta de modelo para pagamentos eletrônicos a partir de plataformas móveis.

1.4 Trabalhos Relacionados

NETO [2] apresenta em seu trabalho um histórico da primeira iniciativa de pagamento móvel no Brasil, a PAGGO - uma iniciativa da empresa de Telecom Oi. É feito também um "survey"sobre nomenclaturas e termos mais comuns do mercado. A PAGGO foi pioneira em serviços de pagamentos, mas o serviço não conquistou o número de Clientes mínimo necessário, levando a Oi a retirar o serviço de seus planos de marketing.

PUJOLLE *et al.* [6] descrevem uma solução de gestão de identidade global baseada em elementos de segurança virtuais, em atendimento ao projeto *SecFuNet*[4]. Nesta solução, é sugerida uma arquitetura de identificação única de usuários centralizando todas as informações necessárias para autenticação, autorização e registro de atividades.

KREUTZ e FEITOSA [15] apresentam em seu trabalho a Gestão de Identidade como serviço (*IdP-as-a-Service*), o que sugere a possibilidade de modelos de negócio globais que tenham como objetivo a administração da identididade de usuários

[3]Será citada a abreviação em inglês, por ser mais recorrente na literatura

[4]SecFuNet Project: "Security for Future Networks", é um projeto conjunto entre Comissão Européia ("7th Framework Programme") e o CNPq.

para diversas modalidades de serviços.

GUTTERMAN *et al.* [16] explica o funcionamento do gerador de números pseudo-aleatórios do Linux e, por consequência, também do Android. A despeito da falta de documentação sobre o assunto, o trabalho detalha o procedimento desde a coleta de entropia do sistema até o algoritmo utilizado para a geração de números pseudo-aleatórios.

DODIS *et al.* [17] demonstram neste trabalho a possibilidade de ataques baseados na baixa entropia de partida dos sistemas baseados em Linux. O trabalho foca no funcionamento de */dev/random* (o dispositivo virtual para geração de números pseudo-aleatórios) e demonstra esta fragilidade.

KAPLAN *et al.* [18] descrevem um ataque a estrutura */dev/random* do Linux utilizando solicitações ICMP (*Internet Control Message Protocol*) IPv6 com bit de fragmentação ativo, tendo por objetivo demonstrar a fragilidade de "reseed"(etapa na qual um gerador pseudo-aleatório de números deve ser novamente reiniciado com entropia do sistema) do PRNG.

DING *et al.* [8] apresentam estudo e demonstrações comprovando que o sistema Linux/Android */dev/random* (PRNG) é capaz de produzir números pseudo-aleatórios com entropia suficientemente alta para atender os requisitos dos protocolos de criptografia, mas também demonstram com outra abordagem o problema da entropia de partida do Linux após *boot* do sistema e possíveis consequências como exploração de *stack buffer overflow* (fragilidade na qual o sistema permite que aplicações não autorizadas se apropriem de informações de memória reservada para outras aplicações) após um ataque contra a entropia.

SCHNEIER [19] é um livro com a descrição teórica e prática dos principais protocolos de segurança utilizados nesta dissertação. Não apenas a criptografia e outras técnicas de proteção da informação é focalizado por *Schneier* neste trabalho, mas também toda a preocupação com a gestão da segurança, a manutenção de procedimentos e preocupação com o elemento humano também são abordados.

FERGUSON e SCHNEIER [20] descrevem neste livro o funcionamento do modelo proposto para criação do PRNG criptograficamente seguro chamado *Fortuna*. A utilização do Fortuna foi uma opção nesta dissertação como contingência dos problemas de entropia no sistema Linux já citados.

KELSEY *et al.* [21] desenvolvem o conceito de *Key stretching* (ou aumento do tamanho da chave utilizada em um sistema de segurança) com a aplicação de SALT: um número pseudo-aleatório gerado a partir da maior entropia possível para aumentar a segurança de um sistema que utiliza senhas pequenas, como o PIN por exemplo.

Pierre L'Ecuyer apresenta em L'ECUYER [22] a descrição teórica dos principais métodos para geração de números pseudo-aleatórios em uso, enquanto que em seu trabalho L'ECUYER e SIMARD [23], discorre sobre métodos estatísticos de testes de aleatoriedade para avaliação de implementações de PRNGs.

SIBLEY *et al.* [24] descrevem as principais características que devem ser observadas em um PRNG e, por consequência, as dificuldades para encontrá-las. Este artigo é extremamente recorrente em diversas bibliografias e um dos mais citados (1361 citações, consultado em Setembro 2014) segundo o *Google Scholar*.

MARKOWSKY [25] descreve brevemente a história recente de utilização de PRNG em diversos sistemas e aplicações. O título "*Sad History*"é justificado pelo próprio autor ao descrever que, por diversas vezes na história, a utilização de PRNGs incorretamente implementados provocou problemas de grandes proporções para corporações e usuários no mundo todo.

BOND *et al.* [26] demonstrou em 2014 a fragilidade do sistema *EMV Chip and PIN* que está se estabelecendo como padrão para transações bancárias baseadas em cartão de crédito. O método demonstra a possibilidade de interceptação (*man in the middle*) de informações críticas sobre a identidade do usuário, permitindo a execução de fraudes diversas.

Michael Roland, em seu trabalhos utilizado nesta dissertação, descreve o embate técnico comercial que ocorre no mercado, em dias atuais, em relação ao elemento de segurança (componente, hardware ou software, que controla todas as diretrizes de segurança em comunicações utilizando NFC). Em ROLAND [27] ele apresenta a arquitetura NFC utilizada em "smartphones"Android, bem como as características do Android para utilização do elemento de segurança virtual - técnica chamada de HCE.

ELENKOV [28] descreve em detalhes arquitetura Android e seus aspectos de segurança, além de reunir em seu livro todas as API's (*Application Programming Interface*) necessárias para implementar segurança em aplicativos desenvolvidos para aquela plataforma.

1.5 Organização do Texto

No capítulo 2 é feita uma revisão bibliográfica sobre o cenário de pagamentos eletrônicos em plataformas móveis, destacando os principais agentes, padrões e tecnologias disponíveis. É feito um estudo sobre plataforma Android e todas as suas possibilidades de segurança, métodos criptográficos, armazenamento e geração de aleatoriedade. Também é feito um estudo geral sobre o estado da arte em criptografia e geração de números pseudo-aleatórios.

No capítulo 3 são apresentadas as técnicas de melhoria para geração de aleatoriedade visando aplicações de pagamentos eletrônicos, baseadas nos estudos sobre as vulnerabilidades da plataforma para geração de aleatoriedade criptograficamente segura. Também é apresentado o modelo para evolução da gestão de identidade dos métodos de pagamentos eletrônicos atuais, seguindo as recomendações EMV para transações baseada na técnica de "tokens".

No capítulo 4 apresentam-se os resultados obtidos a partir das implementações feitas no capítulo 3 avaliando as medidas para comparação com os métodos convencionais.

Por fim, no capítulo 5 são tecidas as considerações finais sobre o modelo de pagamento proposto, uma breve revisão sobre os resultados, contribuições obtidas e perspectivas de trabalhos futuros que poderão ser realizados em virtude das contribuições aqui apresentadas.

Capítulo 2

Revisão Bibliográfica

2.1 Cenários de Pagamentos Móveis

A seguir, serão descritos os principais elementos que compõem um sistema para pagamentos móveis. A nomenclatura utilizada, segue o padrão já mencionado do sistema financeiro (EMV), de forma a manter a integridade com a bibliografia disponível. Como o motivo principal deste estudo é a evolução do sistema financeiro, é importante contextualizar o status atual, tecnológico e processualmente. A proposta nas próximas seções é descrever a topologia das transações financeiras, as tecnologias mais empregadas para realizar estas transações em redes móveis e as tecnologias que estão se mostrando mais viáveis para suportarem uma evolução tecnológica de todo o sistema.

2.1.1 Principais Elementos Envolvidos no Ecossistema

O telefone móvel (MS - *Mobile Sation*, doravante) estabeleceu-se na moderna sociedade como a solução para integração de ferramentas de comunicação e entretenimento. A grande penetração deste serviço em todas as classes sociais faz com que o MS também seja eleito como a grande solução para a gestão de identidade definitiva LERNER [29]. Sendo assim, a utilização do mesmo para viabilizar pagamentos e outras transações financeiras é facilmente percebida – principalmente, devido a evolução do conceito MS para *smartphone*.

Baseado nas definições encontradas em DINIZ *et al.* [30], definem-se os seguintes termos:

Mobile Transactions: Referem-se a operações realizadas através de tecnologias e dispositivos móveis. Adicionalmente aos pagamentos móveis, inclui todo o tipo de transação móvel oferecido pela tecnologia, independente de se envolverem valores financeiros ou não.

Mobile Payment (M-Payment): Os pagamentos móveis incluem pagamentos efetuados ou habilitados por meio de tecnologias de mobilidade digital, através de dispositivos portáteis, com ou sem o uso de redes de telecomunicações móveis. Estes pagamentos são transações financeiras digitais, embora não necessariamente ligados a instituições financeiras ou bancos.

Mobile Banking: Pode ser entendida como um conjunto de serviços bancários móveis, envolvendo o uso de dispositivos portáteis conectados a redes de telecomunicações.

Mobile Money: O dinheiro eletrônico - sendo essencialmente digital.

De acordo com NETO [2], podemos estabelecer os seguintes agentes como imprescindíveis para o funcionamento de um sistema de pagamento, não apenas para pagamentos móveis.

- Cliente: É o titular do meio de pagamento. É ele quem efetua o pagamento ou transação financeira. Nos meios convencionais, é ele o portador do cartão emitido por uma entidade financeira. Sua identidade é a informação diretamente afetada em caso de violações e corrupção do sistema.

- Estabelecimento: É o termo genérico para definição do ponto de venda de algum produto ou serviço. Também mencionada diversas vezes na literatura como Ponto de Venda. Neste ponto ocorre a manifestação de se iniciar uma transação comercial.

- Adquirente: é designação para empresa responsável pelo credenciamento, gerenciamento e relacionamento com os estabelecimentos comerciais e pela intermediação dos fluxos financeiros entre emissores de cartões e os estabelecimentos que aceitam o produto. É quem possui a rede de captura das transações e mantém um acordo contratual com o comerciante (dono do estabelecimento) para processar as transações realizadas através de cartões. O adquirente reembolsa o comerciante pelo valor da compra e cobra uma comissão pelo serviço.

- Bandeiras: São as entidades que emitem concessões de licença para a comercialização do sistema de pagamento eletrônico, convencionalmente cartão. Estas bandeiras podem ser formadas por associações de bancos que atuam mundialmente. O logotipo das bandeiras é estampada no cartão do usuário, sendo responsáveis pela campanha de *marketing* junto ao público visando a divulgação do serviço. Estão ativamente pesquisando novos métodos para a realização das transações bancárias, de modo a permitir a evolução dos sistemas e aumentar a viabilidade financeira dos mesmos (lucros). Está em seu escopo de atuação garantir a integração e uniformidade das informações entre os diversos agentes do sistema. Estão entre as mais conhecidas bandeiras do

mundo estão VISA®, MASTERCARD®, EUROPAY®, todas signatárias do padrão EMV que especifica o padrão para transações financeiras.

- Emissor: São as entidades financeiras que concedem crédito diretamente para o Cliente, o portador do cartão ou titularidade no sistema eletrônico de pagamento. Os cartões podem ser emitidos por bancos ou administradoras. Além de emitir, administram cartões próprios ou de terceiros. Já as administradoras são empresas não financeiras, que como tal não concedem crédito, mas representam os portadores perante às instituições financeiras para a obtenção de crédito. O relacionamento do portador no uso do seu cartão deve ser feito com o emissor, já que é ele quem estabelece os limites desse crédito, e é responsável pelos benefícios do seu cartão (consultas, atendimento, emissão de faturas, programas de milhagem, etc.). Os emissores trabalham em parceria com as "bandeiras", que não são seus concorrentes.

As Figuras 2.1 e 2.2 são ilustrações do ecossistema e agentes (alguns dos mais conhecidos no mercado), conforme descrito anteriormente.

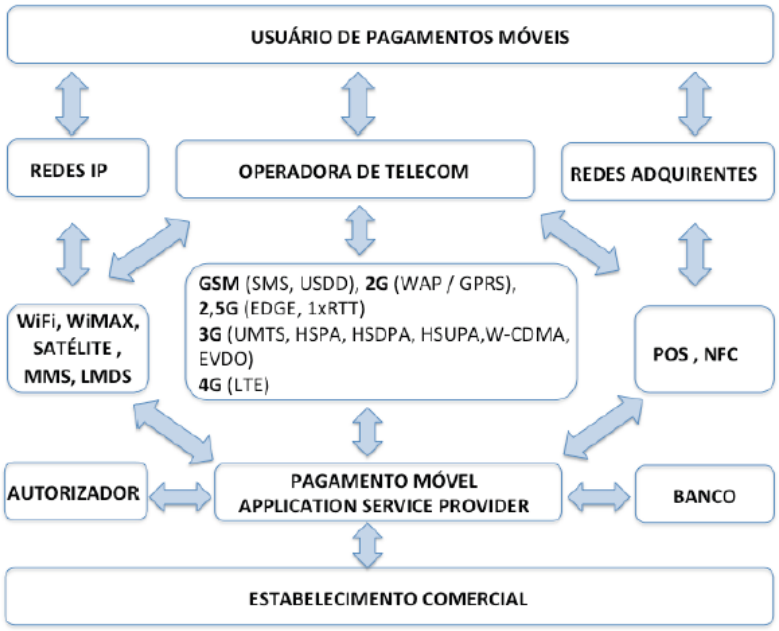


Figura 2.1: NETO [2] Ilustração do ecossistema descrito.

2.1.2 Principais Tecnologias Empregadas

A seguir, será apresentado um "*survey*"das principais tecnologias empregadas nos sistemas mais largamente difundidos hoje.

SMS (*Short Message Service*) 3GPP [31]: Sistema disponível em todos os aparelhos GSM em uso e, por isso, muito utilizado em transações móveis de toda a

Figura 2.2: Ilustração dos agentes, citando alguns dos mais conhecidos no mercado.

natureza. Opera em sistema "*Store and Forward*" e isso dificulta a sua utilização em transações que necessitam resposta imediata – como pagamentos, por exemplo MASSOTH e BINGEL [32]. Segundo NETO [2], no estudo de caso da PAGGO[1] que esta característica "store and forward" foi fator determinante para o fracasso do serviço, dada a percepção de lentidão na comunicação entre Operadora e usuários.

USSD (Unstructured Supplementary Service Data) 3GPP2 [33]: Serviço de comunicação instantânea, também nativo em aparelhos GSM. A comunicação USSD é instantânea e orientada a conexão. A comunicação não ocorre entre usuários diretamente e, sim, entre usuário e uma aplicação (provida pela operadora de serviço móvel). É uma aplicação pouco explorada devido a uma característica do protocolo: a comunicação não é criptografada (nem mesmo pela criptografia GSM padrão GRECAS *et al.* [34], 3GPP [35]. Embora a aplicação tenha grande interesse pelos bancos para serviços de URA (Unidade de Resposta Audível) , a utilização do USSD no Brasil é muito pequena.

NFC (Near Field Communication) FORUM [36]: tecnologia adotada pelo mercado para pagamentos "sem contato"(apenas a aproximação do aparelho móvel). Até final de 2013, ainda existiam dúvidas sobre a continuidade desta tecnologia como solução para pagamentos (especialmente pela relutância da APLLE em adotá-la). Google e outros fabricantes sempre se posicionaram favoráveis a sua adoção, já estando disponível na maior parte de smartphones com sistema Android embarcado. Porém, em Agosto de 2013, Apple discretamente submeteu sua patente de NFC para iOS [13], enviando um (sutil) recado para o mercado que o NFC não teria nenhum questionamento. NFC será abordado em detalhes posteriormente. Por ora, é necessário estabelecer como premissa que nenhuma implementação de

[1]Serviço lançado em 2007 pela operadora Oi: www.oi.com.br

sistemas de pagamentos móveis pode ignorar a utilização posteriori de NFC como protocolo de solução "sem contato".

WAP (Wireless Application Protocol) [14]: este protocolo foi criado para adaptar a navegação WEB de aparelhos antigos, cujas definições gráficas não eram compatíveis com as diversas fontes multimídia na Internet. Porém, com o surgimento dos smartphones e, anteriormente, aparelhos com resolução gráfica melhores e capacidade de navegação html, este protocolo encontra-se em franco desuso sendo totalmente substituído por métodos de acesso via GPRS/3G/LTE. Os testes utilizando WAP para implementar pagamentos eletrônicos mostraram-se muito ineficientes no quesito performance para o usuário final: de todas as implementações, foi a mais lenta na entrega das transações MASSOTH e BINGEL [32].

RFID (Radio Frequency Identification Device) : implementação de "tags" inteligentes, em geral passivos (precisam de fonte externa excitação eletrônica). Para a interface de rádio, ambos NFC e RFID utilizam-se da mesma identificação. A diferença básica entre NFC e RFID é que o primeiro consegue implementar uma comunicação P2P (ponto a ponto entre usuários) com o controle deste enlace implementado no protocolo ISO/IEC 18092:2004 ISO [37]. RFID não é indicado para implementação de pagamentos móveis porque as transações são de natureza complexa, sendo originadas pelo usuário na maioria das vezes.

Implementações em Nuvem: as mais atualizadas implementações de pagamentos eletrônicos exploram fortemente a capacidade dos smartphones para desenvolvimento de aplicações embarcadas (*apps*) e a alta capacidade de dados como 3G e LTE. Será abordado posteriormente a aplicação de elementos virtuais, implementadas a partir de aplicativos Android/iOS ou qualquer outra plataforma móvel. Estas implementações tendem a trazer a solução de pagamentos eletrônicos para o domínio dos provedores de serviços (também conhecidos com *Over The Top* – Google, Yahoo®, Apple, etc.) que desenvolvem as soluções em seus sistemas distribuídos (nuvem) e mantém comunicação com os usuários por meio dos acessos móveis de dados já citados. Esta disputa é tema que será melhor desenvolvido posteriormente.

Em POURALI *et al.* [38] encontra-se um estudo das mesmas tecnologias mais utilizadas. A Tabela 2.1 apresenta um resumo com o "survey"de serviços e as respectivas tecnologias empregadas em pagamentos eletrônicos.

Tabela 2.1: Principais tecnologias empregadas

Method of payment	Business model	Payment technology	Mode of payment	Disadvantages
BOUK[7],[6]	Operator-Centric	SMS	Micro payment	Non macro payment, Non –non repudiation
PayPal[8]	peer-to-peer	SMS,WAP	Micro & Macro payment	Authentication method dialog[4]
pay Box[8]	Bank- Centric	SMS	Macro payment	Non Authentication, Not suitable for micro payment
PayforIT[19],[8]	Operator-Centric	WAP	Micro payment	Non macro payment, High cost in application of WAP technology
Osaifu-Keitai[8]	Operator-Centric	RFID	Micro payment	Non macro payment, Non Payment remote
Google Wallet[9]	Collaboration	NFC	Micro & Macro payment	Non Payment remote
ISIS[10]	Collaboration	NFC	Micro & Macro payment	Non Payment remote
Square Wallet[20]	Collaboration	WAP	Micro & Macro payment	High cost in application of WAP technology
Jiring [11]	Operator-Centric	USSD	Micro mini payment	Non macro payment
Paypaad [12]	Collaboration	SMS,USSD	Micro payment	Non macro payment, Non Authentication
The proposed method	Collaboration	SMS	Micro & Macro payment	

2.1.3 A Evolução do modelo de pagamento por aproximação

NFC (*Near Field Communication*) é um padrão de indústria para comunicação em curtas distâncias (alguns centímetros) que opera na faixa de frequência de 13,56MHz - a mesma faixa da tecnologia RFID (*Radio Frequency Identification Devices*). Está se difundindo e popularizando desde que o Google resolveu inserir no Android bibliotecas e hardware NFC a partir de 2010, NFC iniciou sua popularização junto ao grande público usuário de smartphone.

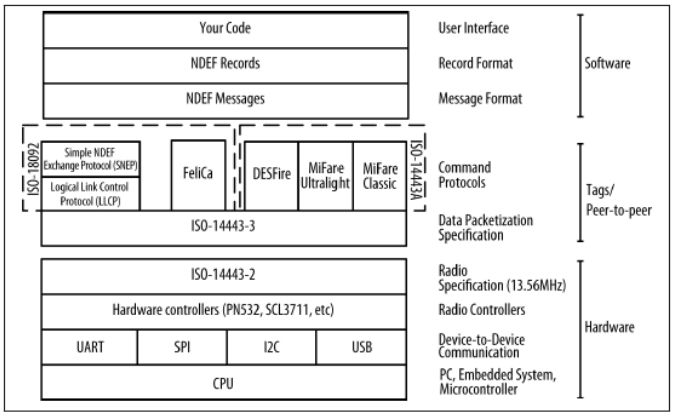
A comunicação NFC é regida pela seguinte pilha de protocolos, de acordo com a tabela 2.2 retirada de IGOE *et al.* [39].

A grande evolução do padrão NFC sobre a especificação de rádio RFID, foi a inserção de um elemento de inteligência para controle de fluxos das comunicações, o SMARTCARD (cartões de identificação equipados com *microchip*, capazes de receberem instruções codificadas). A série ISO/IEC[2] 7816 especifica o funcionamento do *smartcard* de acordo com as divisões abaixo:

- Parte 1: descreve as características físicas dos *smartcards*.
- Parte 2: descreve as interfaces de contato.

[2]Vide tabela de abreviaturas

Tabela 2.2: Protocolos NFC



- Parte 3: descreve a interface elétrica e os protocolos de transporte de camadas baixas.

- Parte 4: descreve o protocolo da camada de aplicação.

A interface física NFC é a mesma utilizada para RFID, ou a série ISO/IEC 14443 de acordo com a distribuição de funções a seguir.

- Parte 1: especifica as características físicas do cartão e antena.

- Parte 2: define a modulação e o esquema de codificação para transmissões binárias e a fonte de alimentação.

- Parte 3: descreve a camada de acesso, com o controle de anti-colisão e o formato de quadro.

- Parte 4: descreve especifica o controle e bloqueio de quadro em protocolo *half-duplex*.

O convívio das duas séries é ilustrada na figura 2.3, conforme ROLAND [27].

Para esta dissertação, o mais importante a destacar no padrão NFC é o aspecto de segurança. Para garantir a segurança na utilização do NFC em situações críticas, como transações financeiras, por exemplo, é especificado um elemento que terá a responsabilidade de armazenar todas as credenciais de segurança (certificados, códigos de identificação, chaves criptográficas para transmissão de identidades, etc.). A este elemento é dado o nome de Elemento de Segurança (ou, como será adotado doravante nesta dissertação, sua abreviação em inglês SE[3]).

O padrão NFC especifica que exista no elemento que o implementa um elemento de segurança (*Security Element* ou SE, para maior coerência com a literatura) que faça o armazenamento de todas as informações de segurança (certificados digitais,

[3]Secure Element

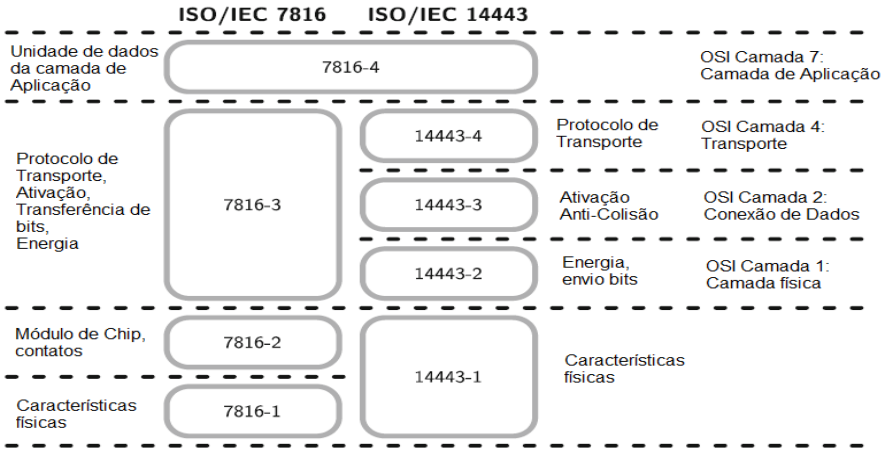


Figura 2.3: Comparação das séries ISO/IEC 7816 e ISO/IEC 14443 na pilha de protocolos NFC.

senhas e chaves criptográficas, etc.). Este elemento em geral é implementado dentro do circuito integrado do próprio chip central controlador NFC. Entretanto, o principal fabricante de chips NFC (NXP *Semiconductors*)[4] desenvolveu uma modalidade de hardware (SmartMX®) que permite a integração do controlador NFC com outros elementos de terceiros. Para isso, especificou duas modalidades de comunicação:

- Comunicação via hardware, sendo o SE igualmente um chip embarcado no equipamento NFC mas com outro fabricante, que não mesmo do chip NFC. Este protocolo de comunicação foi denominado pela NXP por **NFC CWI=Wired Interface** e sua topologia está ilustrada na figura 2.4. Neste caso, a comunicação entre os dois chips ainda atenderá o protocolo ISO/IEC 7816 que determina esta comunicação.

- Comunicação via Software, devido ao interesse de grandes desenvolvedores de serviço como Google que entendem ser mais interessante que estas informações de segurança estejam armazenadas em seu próprio sistema operacional (Android). Desta forma, as empresas de software deverão construir e divulgar suas API's (*Application Programming Interface*) para utilização desta integração. O Google denominou esta modalidade de "SE virtual"como HCE: Host Card Emulation, uma vez que qualquer software aplicativo que manipule estas API's possa realizar as funções de um SE. A figura 2.5 representa esta forma de integração.

[4]http://www.nxp.com/

Figura 2.4: Comunicação NFC-SE através de hardware

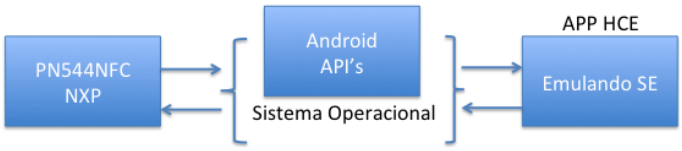


Figura 2.5: Comunicação NFC-SE através de software

2.2 Segurança em Serviços de Pagamentos Eletrônicos

2.2.1 Percepção de Segurança Pelos Usuários de Pagamentos Eletrônicos

Em um levantamento feito no trabalho de Linck, Pousttchi e Wiedemann LINCK *et al.* [40], foram relacionados os aspectos de segurança mais relevante para o usuário final a fim de motivá-lo a adotar uma solução digital em detrimento de uma solução tradicional como cartão de crédito físico (plástico). Primeiramente, foram estabelecidos as propriedades que definem a segurança em transações financeiras, tal qual relacionado na Tabela 2.3. Em seguida, foi questionados a um grupo de pessoas "O que você demandaria para se sentir seguro a utilizar um sistema de pagamento eletrônico?". O resultado segue demonstrado na tabela 2.4. Nesta tabela, foram distribuídas as respostas em agrupamentos de similaridades (foram distribuídas e agrupadas 4998 respostas). Foi demonstrado que a grande preocupação dos usuários recai sobre capacidade de "vazamento"de dados financeiros, que poderiam trazer impactos financeiros imediatos. Esta metodologia vai ao encontro da intuição dos usuários e reação dos mesmos frente as notícias de fraudes e desvios no sistema financeiro. Como conclusão principal, este trabalho apresenta dados empíricos para estabelecer a Segurança como reque-

rimento *sine qua non* para o desenvolvimento de serviços de pagamentos eletrônicos.

Deve-se ressaltar que este *survey* foi realizado utilizando-se uma plataforma digital de questionamentos desenvolvido e mantido pelo MIT (Massachusetts Institute of Technology), o que levou a escolha de público respondente com alto grau de escolaridade e capaz de discernir conceitos sofisticados como criptografia, confidencialidade, dentre outros citados no resultado desta pesquisa. Enfim, pode-se afirmar que o público não representa uma amostragem real e fidedigna do perfil típico dos usuários de um sistema de pagamento móvel. Mas, para o propósito desta dissertação serve como excelente referência inicial dos principais critérios que devem ser considerados no "design"de software com interação direta do usuário final com a plataforma de serviços. Estes critérios devem ser tidos como motivadores para a utilização e fatores de sucesso de um serviço que ainda precisa conquistar espaço na percepção do público com solução alternativa viável ao uso do cartão de crédito convencional.

Tabela 2.3: Propriedades gerais da segurança em Transações Eletrônicos.

Objetivos de Segurança	Definição	Técnica Correspondente para Implementação
Confidencialidade	Propriedade que garante que nenhuma transação será vista por pessoas não autorizadas.	Criptografia
Autenticação	Propriedade que estabelece que as partes envolvidas na transação são exatamente que se presume que sejam no processo.	Posse (do aparelho celular, por exemplo), Conhecimento (PIN, por exemplo) e Característica (biometria, por exemplo).
Integridade	Propriedade na qual a informação na transação ficará intacta durante a transmissão e não será alterada.	Assinatura Digital
Autorização	Propriedade em as partes envolvidas devem estar aptas a verificarem se qualquer um envolvido em uma transação tem permissão para fazê-la.	Assinatura Digital
Não-Repúdio	Propriedade na qual ninguém pode estar habilitado a negar que uma transação foi realizada sem seu conhecimento.	Assinatura Digital

2.2.2 A Criptologia

A Criptologia, segundo SCHNEIER [19], compreende as técnicas utilizadas para a codificação da informação de forma segura e também a recuperação da mesma através de técnicas de decodificação. Ao conjunto de técnicas destinadas a codificação matemática de mensagens é dado o nome de Criptografia. Ao passo que, as técnicas para decodificação e recuperação das mensagens o nome de Criptoanálise. A fig. 2.8 mostra uma visão geral da criptologia, unindo os conceitos de criptografia e criptoanálise.

Tabela 2.4: Critérios mais avaliados pelos usuários de sistemas eletrônicos de pagamento.

Posição	Categoria	Ocorrências	%
1	Confidencialidade	646	12,93
2	Criptografia	611	12,22
3	Segurança “em geral”	586	11,72
4	Transparência e rastreabilidade	578	11,56
5	Autenticação e Autorização	424	8,48
6	Confiança no Provedor de Serviço	413	8,26
7	Proteção contra fraudes	348	6,96
8	Conveniência e Facilidade de uso	298	5,96
9	Infraestrutura Segura	263	5,26
10	Disponibilidade do Sistema	111	2,22
11	Facilidade de Cancelamento	107	2,14
12	Certificação de Terceiros	89	1,78
13	Domínio Técnico	81	1,62
14	Aceitação Ampla no Mercado	76	1,52
15	Anonimato	70	1,40
16	Outros	297	5,94
	Total:	**4998**	**100**

2.2.3 Criptografia:a ciência de escrever seguramente com o objetivo de esconder o conteúdo de uma mensagem

Em tempos atuais, o termo criptografia está muito associado a algoritmos computacionais e sofisticados processos envolvendo alta tecnologia. Mas, como citado no item anteriormente, a prática de esconder mensagens já data de séculos. Em 2000 A.C., os gregos se utilizavam de um instrumento chamado *Scytale* para a codificação de mensagens (Figura 2.6), o que posteriormente também ficou conhecido como ”cifrador de César”PAAR e PELZL [3].

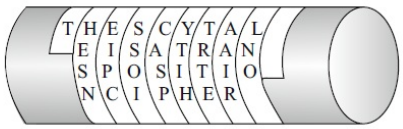


Figura 2.6: PAAR e PELZL [3] *Scytale* de Esparta.

Durante a segunda grande guerra, os alemães desenvolveram um dispositivo chamado *Enigma*, através do qual eles conseguiam enviar mensagens telegrafadas devidamente codificadas, assegurando que somente os detentores deste equipamento (tropas alemãs) pudessem decodificar as informações (Figura 2.7). Sabe-se que o episódio de captura destes equipamentos pelas tropas aliadas representam uma grande reviravolta no desenvolvimento do conflito.

O estudo da Criptografia requer a análise das diversas ferramentas para a codificação de mensagens, bem como a capacidade de ”quebrar”esta codificação. Esta análise pode levar ao estabelecimento de limites para a utilização de determinado

Figura 2.7: PAAR e PELZL [3] Enigma: equipamento alemão utilizado na 2ª Guerra para embaralhar mensagens.

método.

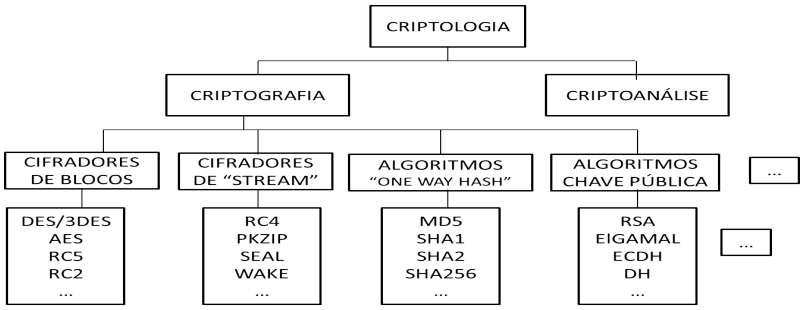


Figura 2.8: Diversos ramos da Criptologia.

2.2.4 Esteganografia

Outra técnica empregada ao longo do tempo para esconder informações de pessoas não autorizadas é a Esteganografia. A esteganografia não é uma área nova: muitas técnicas desta área são conhecidas há milhares de anos. Na Grécia Antiga, por exemplo, tabletes de madeira cobertos com cera eram utilizados para escrita e comunicação - as informações eram escritas na cera, e quando elas não eram mais necessárias, a cera era derretida e uma nova camada era colocada sobre a madeira. Na Segunda Guerra Mundial, as tintas invisíveis começaram a ser utilizadas. As mensagens escritas por esse tipo de tinta só poderiam ser lidas se o papel fosse aquecido.Hoje em dia, a esteganografia digital vem sendo frequentemente utilizada. Nela, meios digitais, como imagens, arquivos de áudio e a Internet, são usados para esconder mensagens secretas SCHNEIER [19]. Ao contrário do que pode parecer, esteganografia e criptografia são duas áreas com objetivos diferentes. Enquanto o segundo tem o propósito de impedir que as pessoas saibam o conteúdo de uma

mensagem, o primeiro se baseia em evitar que as pessoas saibam que a mensagem existe. Ou seja, na criptografia, os receptores sabem da existência das mensagens, porém não conseguem, a princípio, lê-las; a esteganografia tenta fazer com que os receptores não percebam que há uma mensagem naquele meio. Agências militares e de inteligência precisam de meios discretos para se comunicar, sobretudo em áreas de conflito. A transmissão de conteúdo criptografado não é muito eficiente neste quesito, dado que o emissor do sinal pode ser facilmente localizado e possivelmente atacado. Por este motivo, técnicas de esteganografia são largamente usadas em comunicações militares, como a modulação por espalhamento de espectro, dificultando a detecção da transmissão pelo inimigo.

Em PETITCOLAS *et al.* [4], é apresentada uma proposta de taxonomia para as diversas formas de ocultação de informação (Figura 2.9). Além da já citada esteganografia, temos outras técnicas amplamente divulgadas e empregadas em estratégias comerciais atualmente:

1- *Watermarking*: inserção de um identificador (visível ou não) em documentos para impedir sua duplicação ou cópia não autorizada (Figura 2.10).

2- Ruído: é uma técnica simples que consiste em substituir o ruído em uma imagem ou em um arquivo de áudio pela informação que se deseja transmitir;

3- Espalhando a Informação: mecanismos mais sofisticados espalham a informação nos pixels de uma imagem ou em partes de arquivos de áudio;

4- Ordenação: consiste em transmitir a informação através da ordem em que os elementos de uma lista são dispostos;

5- Dividindo a Informação: divide a mensagem em partes que seguem caminhos diferentes até o destino; algumas técnicas mais sofisticadas possibilitam, inclusive, que a informação seja reconstruída a partir de uma fração do total de pacotes em que a mensagem foi dividida.

2.2.5 Definições de Segurança

Na moderna literatura sobre segurança e, sobretudo, criptografia, faz-se necessário padronizar as principais terminologias empregadas na descrição de um sistema que emprega técnicas de segurança da informação. Especialmente, os principais agentes do processo (sejam automáticos ou elementos humanos).

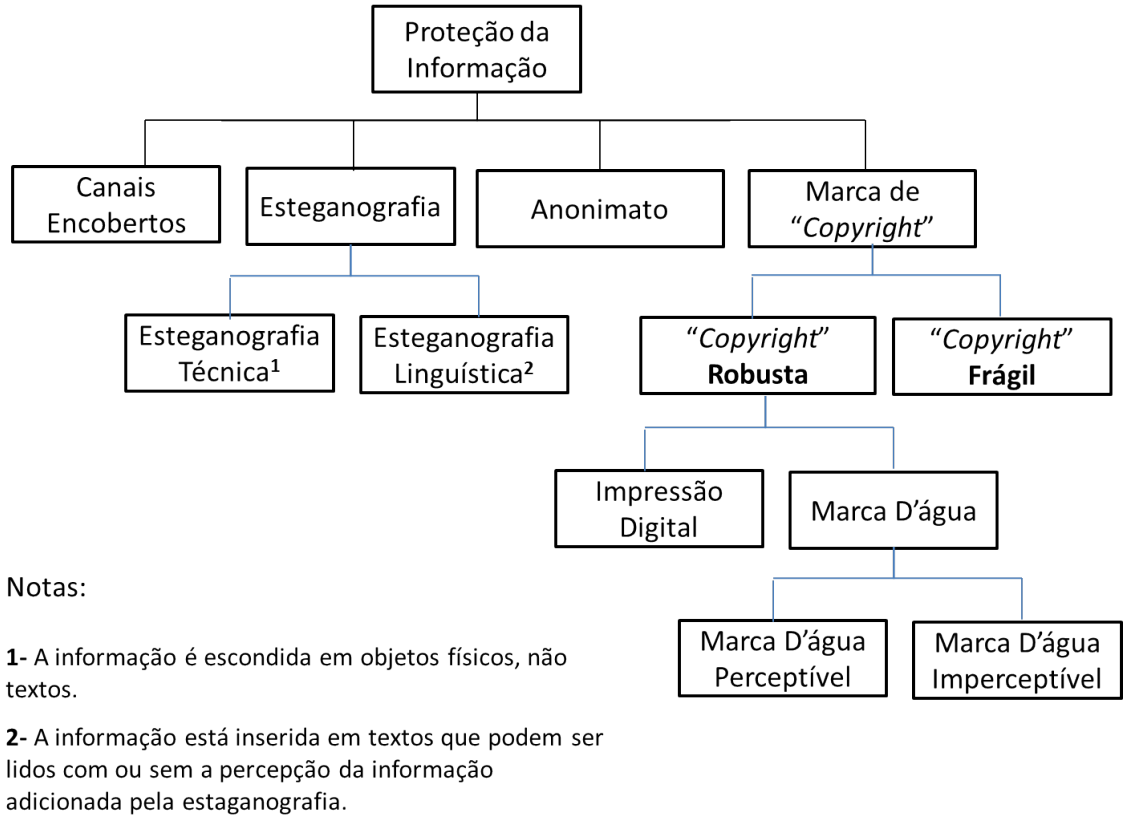


Figura 2.9: PETITCOLAS *et al.* [4] Classificação de técnicas para ocultação de informação.

Elementos básicos em um sistema de comunicação segura

- Emissor: responsável por gerar a mensagem que se deseja transmitir.
- Receptor: destinatário da mensagem, o qual estará apto para entender a mensagem tão logo lhe seja entregue.
- Codificador/Decodificador: solução geralmente baseada em criptografia que utilizará uma chave (de conhecimento tanto do emissor quanto do receptor) para criptografar a mensagem.
- Canal seguro: meio pelo a mensagem irá ser transmitida "apenas"após o processo criptografada.
- Canal inseguro: meio pelo qual (pode ser, fisicamente o mesmo que o anterior) as mensagens poderão trafegar sem proteção de criptografia.
- Chave: será a sequência numérica utilizada pelo algoritmo de criptografia para os processos de codificação e decodificação. Somente emissor e receptor terão acesso a esta chave ou a conhecimento de como gerá-la.

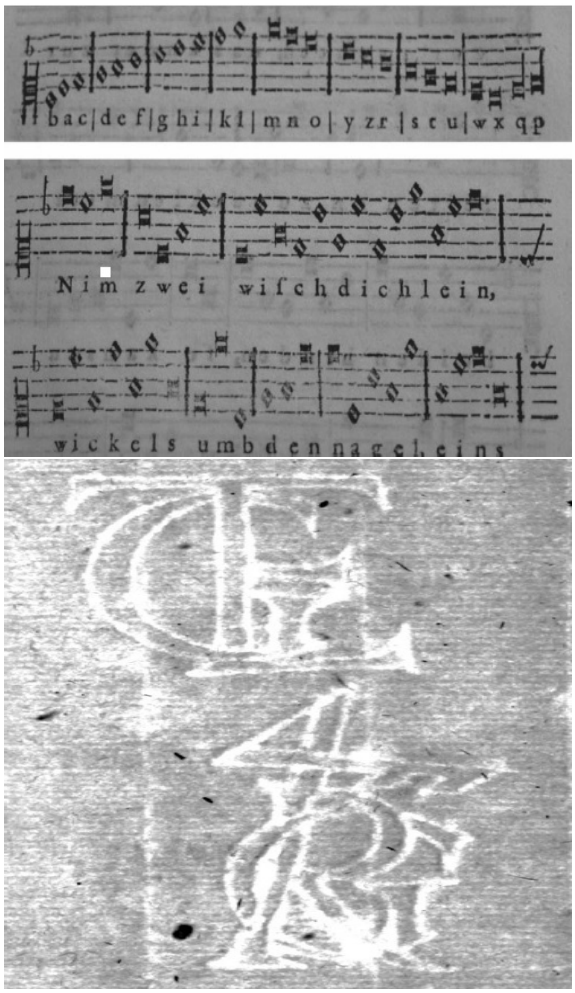


Figura 2.10: PETITCOLAS *et al.* [4] Exemplos de *watermarking*: mapeamento de letras em partitura musical (figura superior) e uma das marca d'águas mais antigas (1550) da Inglaterra (figura inferior).

- Atacante: será o elemento cuja intenção é sobrepujar o sistema de segurança e acessar, sem autorização, o conteúdo da mensagem. Para isso, este elemento irá desenvolver inúmeras modalidades de ataques sobre o meio não protegido (como será visto posteriormente).

Como citado anteriormente, o conhecimento da chave exclusivamente pelos emissor e receptor é garantia de sucesso do processo.
Na obtenção de confidencialidade ou a garantia que o conteúdo da mensagem que se deseja proteger será devidamente acessado somente por agentes autorizados, deve-se buscar, através das soluções a serem adotadas no canal seguro, obter garantias que a autenticação, integridade e não repúdio possam estar presentes no processo de gestão da informação. De acordo com SCHNEIER [19], podem-se definir estes elementos da seguinte forma:

- Autenticação: garantir que as identidades do emissor e receptor da mensagem

possam ser atestadas, de modo que um intruso na comunicação não consiga assumir indevidamente uma destas identidades.

- Integridade: deve ser possível ao receptor da mensagem verificar se ela foi modificada durante o processo de transmissão; um intruso não pode estar apto a trocar uma mensagem original por outra falsa.

- Não Repúdio: um emissor não pode, em hipótese alguma, negar a autoria de uma mensagem originada por ele após o mesmo emitir esta mensagem.

Estes três conceitos irão nortear a implementação de todos os sistemas de segurança tanto para redes móveis quanto para qualquer outro fim. A garantia destes três conceitos será de suma importância aos administradores da rede em questão, por que a maioria dos ataques (a serem vistos posteriormente neste trabalho) visarão explorar fragilidades nestes aspectos da comunicação: se apropriar indevidamente do conteúdo assumindo uma falsa identidade, alterando o conteúdo original e depois negando o processo SCHNEIER [41].

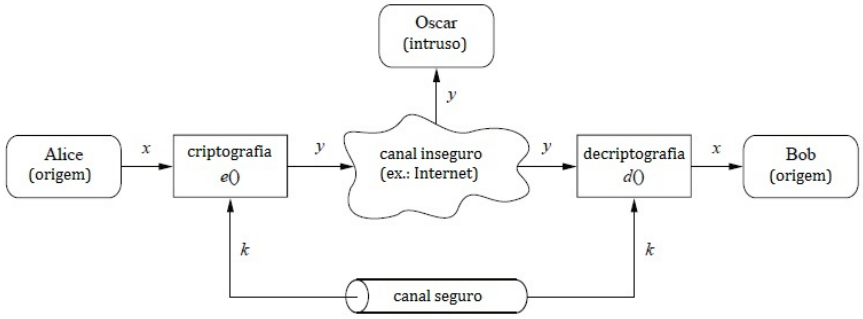


Figura 2.11: PAAR e PELZL [3] Estrutura básica de um sistema de segurança.

2.2.6 Introdução ao processos de Criptografia

A criptografia é uma função matemática usada para codificação e decodificação da mensagem SCHNEIER [19]. As primeiras implementações matemáticas e com a utilização de algoritmos computacionais, visavam restringir o conhecimento deste algoritmo, de modo que apenas os gestores do processo teriam acesso a estrutura deste algoritmo. Apesar desta estrutura garantir uma boa qualidade na gestão destes algoritmos, pensando em larga escala e principalmente no uso comercial da criptografia, este modelo mostra-se inviável - principalmente, após a "explosão"de consumo dos produtos de conectividade em rede, liderados pela Internet. Ou seja, o modelo era viável em aplicações de baixa segurança com baixa demanda de consumo, obviamente não poderia atender as expectativas após o crescimento de aplicações em rede de escala global.

A moderna criptografia resolveu este problema concentrando a segurança não mais na confidencialidade do algoritmo, mas sim na **chave** - doravante, denominada K. Esta chave deve ser qualquer combinação de valores de grande número. O conjunto de todos os valores possíveis desta chave é denominado "espaço de chave", ou *Keyspace* SCHNEIER [19]. Uma melhor definição para o processo segue abaixo. Considere-se M a mensagem a ser protegida no processo:

$$E_K(M) = C \tag{2.1}$$

$$D_K(C) = M \tag{2.2}$$

Aqui vale uma padronização na nomenclatura utilizada neste trabalho: serão adotadas as expressões abaixo para representar cada um dos termos no processo de criptografia:

1- criptografar: ato de utilizar uma solução de criptografia para proteger uma mensagem.

2- decriptogragar: ato de recuperar a mensagem original uma vez que ela tenha passado por um processo de criptografia.

Posto isso, pretende-se evitar confusões de termos como "cifrar", "decifrar", encriptar, etc.
A função 2.1 representa a criptografia da mensagem M. A função 2.2 representa a sua decriptografia. As duas funções possuem a propriedade seguinte:

$$D_K(E_K(M)) = M \tag{2.3}$$

Esta propriedade 2.3 está representada na Figura 2.11, através da qual a mensagem M utilizará de chaves na emissão e recepção (conforme será visto posteriormente, as chaves poderão ser iguais ou não, de acordo com o algoritmo adotado). A confidencialidade do processo passou a residir na capacidade de manter estas chaves em segredo, sem que indivíduos não autorizados tenham acesso às mesmas.

Tipos de Chaves

Como visto em SCHNEIER [19] há duas formas gerais de algoritmos baseados em chaves: simétricos e assimétricos.
Algoritmos Simétricos: chamados de convencionais, a chave para criptografar dados pode ser calculada a partir da chave para decriptografar ou *vice-versa*. Na maioria destes algoritmos, ambas as chaves são iguais. A representação matemática

deste tipo de algoritmo é a mesma vista nas funções 2.1 e 2.2, onde K será a representação da mesma chave nos dois lados do canal de comunicação.

Algoritmos Assimétricos ou Chaves Públicas: assim chamados pelo fato das chaves serem diferentes, além de que a chave para criptografar não pode ser calculada a partir da chave para decriptografar ou *vice-versa*. Estes algoritmos são chamados de "chaves públicas"uma vez que a chave de codificação pode ser de conhecimento público. Um completo estranho pode utilizar esta chave para criptografar mensagens, mas somente uma pessoa específica poderá decriptografar a mensagem, pois ele é o único detentor da chave para isso chamada "chave privada". A representação matemática deste modelo assimétrico segue abaixo:

A criptografia utilizando chave pública será representada por

$$E_K(M) = C \tag{2.4}$$

embora as chaves pública e privada sejam diferentes, a decritografia será representada por

$$D_K(C) = M \tag{2.5}$$

Notar que 2.1 e 2.4 são as mesmas equações, da mesma forma que 2.2 e 2.5 também, mas não deve ser esquecido que as chaves são diferentes.

Por vezes, as mensagens serão criptografadas com a chave privada e decriptografadas com a pública; isso é utilizado em assinaturas digitais (a ser definida posteriormente). A despeito da possibilidade de confusão, estas operações são representadas, respectivamente:

$$E_K(M) = C$$

$$D_K(C) = M$$

Um problema que surge naturalmente é a troca segura de chaves: quando dois interlocutores pretendem estabelecer uma comunicação segura utilizando algoritmos de chaves simétricas ou assimétricas, como fazer com que as chaves sejam compartilhadas de forma segura? Mais detalhes destes mecanismos serão apresentados no item 2.2.11, pois irão demandar um embasamento matemático que será feito no item 2.2.7.

Considerações sobre chave simétrica: o grande desafio é a propagação da chave até o receptor, o que deverá ocorrer em canal seguro *offline* ou utilizando conteúdo criptografado. Algoritmos de chave simétrica são mais rápidos e exigem menos recursos computacionais.

Considerações sobre chave assimétrica: os algoritmos conseguem resolver o problema do transporte de chaves porque existe uma etapa para geração de segredos

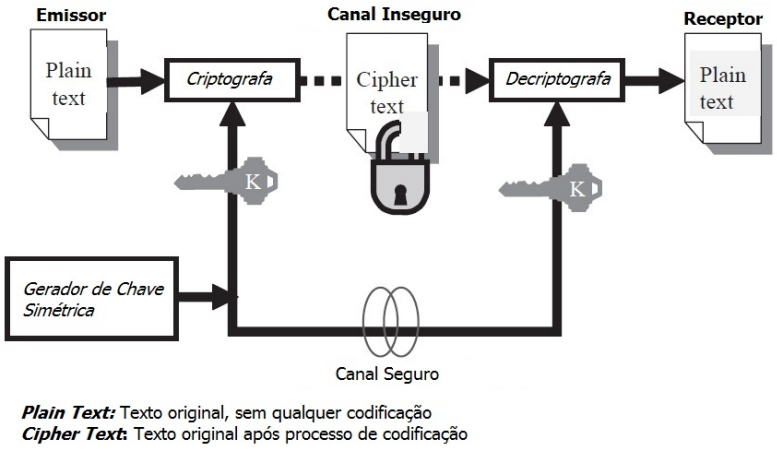


Figura 2.12: CAYIRCI e RONG [5] Utilização de Chave simétrica

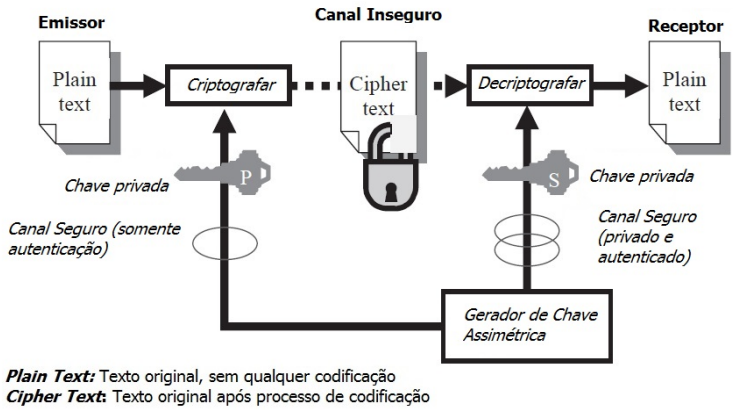


Figura 2.13: CAYIRCI e RONG [5] Utilização de Chave Assimétrica

entre emissor e receptor. Entretanto, estes algoritmos são baseados em segurança utilizando grandes números primos (item 2.2.11) e provocam muito *overhead* de informação a ser transmitida. Devido a isso exposto e também ao fato de demandam mais recursos computacionais para realizarem a criptografia, a solução de chave assimétrica é mais utilizada para assinaturas digitais (item 2.2.8) ou criptografar a chave simétrica que será transportada.

Fica claro com as ilustrações acima que, se um intruso tiver o conhecimento de qual algoritmo está sendo utilizado, bem como a chave k definida pelas duas partes comunicantes, ele será plenamente capaz de interceptar a mensagem e influenciar na comunicação que se pretendia fazer em segredo. Portanto, fica evidente a necessidade de armazenamento seguro da chave utilizada a fim de evitar qualquer possibilidade de violação do segredo considerado. A pergunta consequente dado este cenário é: quanto ao algoritmo de codificação utilizado, deve-se mantê-lo em segredo também?

O princípio de Kerckhoffs

Ao final do século 19, Auguste Kerckhoffs defendeu a ideia contrária a este entendimento de manter o método de criptografia sob segredo, alterando a percepção militar da época, no que veio a ser conhecido como princípio de Kerckhoffs:

> *O método de criptografia não deve ser entendido como um segredo e ele deve estar disponível facilmente para o inimigo, sem que este fato represente qualquer inconveniência.*

Em outras palavras, o sistema de segurança deve funcionar ainda que um adversário esteja de posse do mesmo algoritmo utilizado para a codificação da mensagem e em posição de facilmente interceptar a comunicação. Posto em termos mais práticos, a segurança do sistema não deve estar sob o algoritmo utilizado mas, sim, sob a garantia que a chave esteja armazenada de forma segura e de conhecimento apenas das partes autorizadas na comunicação.

De acordo com KATZ e LINDELL [42], existem 03 argumentos principais em favor do princípio de Kerckhoffs. O primeiro é significantemente mais simples armazenar de forma segura uma chave do que manter um algoritmo em segredo. Isso é mais facilmente identificado na perspectiva de uma grande empresa contemporânea, com milhares de empregados desejando utilizar comunicação segura entre eles mesmos para compartilhar informações confidenciais para o negócio da empresa. A gestão dos algoritmos utilizados por cada par tornar-se-ia extremamente complexa para que a organização o mantenha em segredo, contra a real possibilidade de que algum empregado venha a vazar esta informação.

O segundo argumento recai sobre um caso em que uma das partes divulga incorretamente o algoritmo utilizado para algum terceiro interessado em interferir nesta comunicação segura. É tão mais fácil para as partes envolvidas trocarem a chave em comparação com a substituição do algoritmo empregado. A geração de uma nova sequencia pseudoaleatória como nova chave pode ser feito de forma simples e empregando técnicas seguras.

Finalmente, o desenvolvimento em larga escala de uma aplicação que se utilize de criptografia é bem mais fácil considerando que todo os usuários utilizarão o mesmo algoritmo (com diferentes chaves). Pressupõe-se, neste caso, maior viabilidade técnica para homologação de um algoritmo, correção de erros e atualização do software utilizado.

2.2.7 Embasamento Matemático

Shannon

SHANNON [43] elaborou um trabalho que foi definitivo para a construção matemática da criptoanálise. Ele partiu do fato que toda linguagem possui padrões estatísticos na ocorrência dos símbolos que a compõem (ex.: letras do alfabeto). Tendo em vista que se tratam de símbolos discretos, é possível criar regras para a substituição dos mesmos de forma que se possa reverter o processo posteriormente. Sendo M a mensagem que se deseja codificar e K a chave utilizada, dentre o conjunto possível de chaves, a ação de criptografia (geração do criptograma E) e decriptografia seria escrita como:

$$E = f(M, K) \quad ou \quad E = T_i M$$

$$E = f^{-1}(M, K) \quad ou \quad E = T_i^{-1} M$$

sendo T_i a definição da Transformação T, utilizando a chave K_i aplicada à mensagem M. Da mesma forma, T_i^{-1} corresponde ao inverso da função - ou o processo de decriptografia.

- Entropia da mensagem $H(M)$: quanto maior a quantidade mínima necessária de símbolos para a representação de um conteúdo, pode-se dizer que maior é sua Entropia. Shannon verificou que: cada mensagem M do espaço amostra de mensagens possui uma probabilidade a priori e, consequentemente, cada criptograma E gerado a partir destas mensagens possui uma probabilidade a posteriori associada. Se o criptoanalista "inimigo"estiver de posse de um criptograma formado por N símbolos, quanto maior o valor de N maior a probabilidade de que somente uma mensagem esteja associada àquele criptograma. A entropia de um sistema de criptografia de chave K bits pode ser obtida por:

$$H(K) = \log_2 K \tag{2.6}$$

- Confusão e Difusão SCHNEIER [19]: Confusão é definido como a dificuldade de se fazer uma associação entre o conteúdo original e o criptograma gerado . Difusão é a possibilidade de se dissipar a redundância entre o conteúdo original e o criptograma gerado.

- Sistema Perfeito: dado que cada mensagem M do conjunto possível de mensagens possui uma probabilidade a priori p associada, tem-se do teorema de Bayes:

$$Pr[M|E] = \frac{Pr[M]Pr[E|M]}{Pr[E]} \tag{2.7}$$

onde:
$Pr[M]$ é a probabilidade a priori de M
$Pr[E]$ é a probabilidade de se obter E ao interceptar a comunicação
$Pr[M|E]$ é a probabilidade a posteriori da mensagem M se o criptograma E for interceptado.
$Pr[E|M]$ é a probabilidade condicional do criptograma E se a mensagem M é escolhida pelo criptoanalista - ou seja, a soma de todas as probabilidades individuais de cada chave K as quais produzem o criptograma E a partir da mensagem M.

Para um sistema de criptografia perfeito, $Pr[M|E]$ deve ser igual a $Pr[M]$ para todo E e toda M. Portanto, excluindo a possibilidade de $Pr[M] = 0$,

$$Pr[E|M] = Pr[E]$$

$$Pr[M|E] = Pr[M]$$

Criptografia Perfeitamente Segura

Prosseguindo para a obtenção de uma definição matemática de um esquema de criptografia e, combinando as equações 2.3 e 2.7, será estabelecida a definição utilizada ao longo desta dissertação. Os termos Perfeitamente Seguro (normalmente associado a uma mensagem criptografada) e Segurança Perfeita (normalmente associado a algoritmo ou esquema de criptografia) são empregados com grande frequência em textos sobre segurança mas poucas vezes definidos.

Definição de Criptografia Perfeitamente Segura:

Um esquema de criptografia que utilize um gerador de chave qualquer pertencente ao conjunto K de chaves possíveis, um algoritmo que executa uma função de criptografia Enc e um algoritmo que executa uma função de decritografia Dec, com um espaço amostral de mensagens possíveis $\mathcal{M}$ é tido como perfeitamente seguro se, para toda distribuição de probabilidade de $\mathcal{M}$, cada mensagem $m \,\epsilon\, \mathcal{M}$ e cada texto cifrado c gerado dentro do espaço $\mathcal{C}$ $(c \,\epsilon\, \mathcal{C})$ com $Pr[C = c] > 0$:

$$Pr[M = m \,|\, C = c] = Pr[M = m]$$

Tendo-se assumido uma distribuição probabilística M qualquer para $\mathcal{M}$ e C para $\mathcal{C}$.

O requerimento de $Pr[C = c] > 0$ é preventivo, para não permitir condicionar a um evento cuja probabilidade seja 0.
Desta forma, pode-se prosseguir com a definição de Segurança Perfeita. Para a seguinte formulação matemática, assume-se a independência entre a distribuição de probabilidade do texto cifrado e a distribuição de probabilidade do texto original. Ou, posto de uma forma mais estruturada, para quaisquer duas mensagens m, $m' \epsilon \mathcal{M}$ a distribuição do texto cifrado quando m for criptografada deve ser *i.i.d.* (*independente e identicamente distribuída*) em relação a distribuição do texto cifrado quando m' for criptografada. Formalmente, para cada m, $m' \epsilon \mathcal{M}$, e cada $c \epsilon \mathcal{C}$,

$$Pr[Enc_k(m) = c] = Pr[Enc_k(m') = c] \tag{2.8}$$

Onde as probabilidades estão associadas a escolha de $k \epsilon \mathcal{K}$ e qualquer método de geração de aleatoriedade por *Enc* (o algoritmo gerador de criptografia).
Isso implica que o texto cifrado não contém nenhuma informação sobre o texto original e que é matematicamente impossível distinguir o resultado da criptografa de m do resultado obtido a partir de m', uma vez que as distribuições probabilísticas do texto cifrado são *i.i.d.* para ambas. Posto isso, pode-se partir para uma definição mais abrangente de um processo de criptografia, baseado na leitura de KATZ e LINDELL [42]:

Um esquema de criptografia (Gen, Enc, Dec) com espaço amostral de mensagens $\mathcal{M}$ consegue implementar Segurança Perfeita se e somente se a equação 2.8 for válida para toda e qualquer mensagem m, $m' \epsilon \mathcal{M}$ e todo e qualquer $c \epsilon \mathcal{C}$.

Prova da definição acima: foram estabelecidas as condições para que um esquema de criptografia seja considerado perfeitamente seguro. Fixando-se uma distribuição de probabilidade M para $\mathcal{M}$ e C para $\mathcal{C}$, utilizando-se uma mensagem $m \epsilon \mathcal{M}$ qualquer e um texto cifrado qualquer c para a qual $Pr[C = c] > 0$. Se $Pr[M = m] = 0$ então pode-se obter trivialmente

$$Pr[M = m \,|\, C = c] = 0 = Pr[M = m]$$

Assumindo agora que $Pr[M = m] > 0$, tem-se que

$$Pr[C = c \,|\, M = m] = Pr[Enc_K(M) = c] \,|\, M = m, K = k] = Pr[Enc_k(m) = c]$$

Sendo que a primeira igualdade é por definição a variável aleatória C, e a segunda é função da condição de que M é igual a m e K igual a k. Considere-se $\delta_c \stackrel{\text{def}}{=} Pr[Enc_k(m) = c] = Pr[C = c \,|\, M = m]$. Se as condições da definição de Segurança Perfeita acima estiverem atendidas, então para toda $m' \epsilon \mathcal{M}$ tem-se que

$Pr[Enc_k(m') = c] = Pr[C = c \mid M = m'] = \delta_c$. Usando Bayes novamente, tal como em 2.7, pode-se chegar a

$$\begin{aligned} Pr[M = m|C = c] &= \frac{Pr[C = c|M = m] \,.\, Pr[M = m]}{Pr[C = c]} \\ &= \frac{Pr[C = c|M = m] \,.\, Pr[M = m]}{\sum_{m' \epsilon \mathcal{M}} Pr[C = c|M = m'] \,.\, Pr[M = m']} \\ &= \frac{\delta_c \,.\, Pr[M = m]}{\sum_{m' \epsilon \mathcal{M}} \delta_c \,.\, Pr[M = m']} \\ &= \frac{Pr[M = m]}{\sum_{m' \epsilon \mathcal{M}} Pr[M = m']} \;=\; Pr[M = m], \end{aligned}$$

Onde o somatório em $m' \epsilon \mathcal{M}$ com $Pr[M = m'] \neq 0$. Conclui-se que para cada $m \,\epsilon\, \mathcal{M}$ e $c \,\epsilon\, \mathcal{C}$ para o qual $Pr[C = c] > 0$, atende $Pr[M = m \mid C = c] = Pr[M = m]$, demonstrando portanto que este esquema é perfeitamente secreto.

Criptografia One-Time Pad

Em 1917, Gilbert Vernam SCHNEIER [19] concebeu um sistema de criptografia que utiliza um conjunto infinito de senhas (conjunto de letras), de modo que nenhum símbolo jamais se repetiria durante o processo de codificação. Este sistema ficou conhecido como *One-Time Pad*[5], ou Chave de Uso Único (tradução aproximada), porque cada letra utilizada na chave só ocorre uma única vez. Aplicando um cifrador *Vigenere* (item 2.2.7 adiante) com período 1 (ou seja, o agrupamento de letras da chave tem período d de repetição d=1, então:

$$e_i = m_i + k_i \,(\mathrm{mod} 26)$$

Exemplo: dada a chave TBFRGFARFM e a mensagem ONETIMEPAD, tem-se como saída:

$$\begin{aligned} O + T \,(\mathrm{mod} 26) &= I \\ N + B \,(\mathrm{mod} 26) &= P \\ E + F \,(\mathrm{mod} 26) &= K \end{aligned}$$

A sequência de saída codificada seria IPKLPSFHGQ.

Conforme introduzido no subitem anterior, 25 anos depois de Vernam, Shannon introduziu a definição matemática da perfeita criptografia demonstrando que *One-Time Pad* atendia os requisitos de segurança necessários. A seguir, os passos para a criação da senha perfeita.

[5]Será mantido o termo original em Inglês para fins de coerência com a bibliografia utilizada e amplamente divulgada sobre o assunto.

Fixe um inteiro $\ell > 0$. Sejam o espaço amostral das mensagens $\mathcal{M}$, o espaço amostral das chaves $\mathcal{K}$ e o espaço de texto cifrado $\mathcal{C}$ todos iguais a $\{0,1\}^\ell$ (o conjunto de todas as sequências binárias de tamanho ℓ).

- *Gen*: o algoritmo gerador de chaves escolhe uma chave de $K = \{0,1\}^\ell$, de acordo com a distribuição uniforme (ou seja, cada 2^ℓ independente do espaço K é escolhida como a chave com probabilidade exata $2^{-\ell}$).
- *Enc*: dada a chave $k \epsilon \{0,1\}^\ell$ e a mensagem $m \epsilon \{0,1\}^\ell$, a saída do algoritmo de criptografia será o texto cifrado $c = k \oplus m$.
- *Dec*: dada a chave$k \epsilon \{0,1\}^\ell$ e o texto cifrado $c \epsilon \{0,1\}^\ell$, o algoritmo de decriptografia gera a mensagem $m = k \oplus c$.

Na descrição acima para implementação da criptografia perfeita, entende-se a função entre duas sequências binárias $a \oplus b$ como sendo uma função binária OR-exclusiva (XOR) - a ser mais detalhada na próxima seção. Sendo duas sequências binárias de tamanho ℓ, então entende-se que $a = a_1, ..., a_\ell$ e $b = b_1, ..., b_\ell$. Desta forma, $a \oplus b$ é dado por $a_1 \oplus b_1, ..., a_\ell \oplus b_\ell$.

No esquema de criptografia *one-time pad*, a chave é uma sequência uniforme do mesmo tamanho (quantidade de bits) que a mensagem. O texto cifrado é computado por simples função XOR entre a chave e a mensagem.
Antes da definição matemática da segurança obtida por este esquema de criptografia, deve ser verificado que: $Dec_k(Enc_k(m)) = k \oplus k \oplus m = m$, portanto o esquema *one-time pad* representa um esquema de criptografia válido.

Pode-se estabelecer uma prova do esquema *one-time pad* acima partindo das definições estabelecidas para Criptografia Perfeitamente Segura, bem como da premissa de independência do texto cifrado e a mensagem original - ou seja, não importa qual mensagem $m \epsilon \mathcal{M}$ seja criptografada, o texto cifrado resultante é uniformemente distribuído[6] em $\mathcal{C}$.
Com isso, apresenta-se a seguir a definição formal de *One-Time Pad* e, na sequência, a prova matemática da mesma.

O esquema de criptografia One-Time Pad é perfeitamente seguro.

[6]Ou seja, todos os elementos c que compõem e espaço amostra $\mathcal{C}$ possuem a mesma probabilidade de ocorrência. Para mais informações sobre Distribuição de Probabilidade Uniforme, verificar em Leon-Garcia, Alberto. *Probability, statistics, and random processes for electrical engineering* , 3rd ed, Pearson Education, Inc. - cap.4, pp 163.

Prova da definição acima: Primeiramente, estabelece-se $Pr[C = c|M = m']$ para um arbitrário valor de $c \epsilon \mathcal{C}$ e $m' \epsilon \mathcal{M}$. Assim, para o esquema de senha de único uso discutido,

$$\begin{aligned} Pr[C = c|M = m'] &= Pr[Enc_k(m') = c] = Pr[m' \oplus k = c] \\ &= Pr[k = m' \oplus c] \\ &= 2^{-\ell} \end{aligned}$$

Onde a igualdade final é determinada pelo fato de $K = k$ é uma sequência de ℓ-bits (uniformemente distribuída sobre o espaço amostra $\mathcal{K}$). Fixando qualquer distribuição M sobre $\mathcal{M}$, para qualquer $c \epsilon \mathcal{C}$, tem-se que

$$\begin{aligned} Pr[C = c] &= \textstyle\sum_{m' \epsilon \mathcal{M}} Pr[C = c|M = m'] \centerdot Pr[M = m'] \\ &= 2^{-\ell} \centerdot \textstyle\sum_{m' \epsilon \mathcal{M}} Pr[M = m'] \\ &= 2^{-\ell}, \end{aligned}$$

onde o somatório sobre $m' \epsilon \mathcal{M}$ é definido para $Pr[M = m'] \neq 0$. Utilizando novamente o teorema de Bayes:

$$\begin{aligned} Pr[M = m|C = c] &= \frac{Pr[C = c|M = m] \centerdot Pr[M = m]}{Pr[C = c]} \\ &= \frac{2^{-\ell} \centerdot Pr[M = m']}{2^{-\ell}} \\ &= Pr[M = m] \end{aligned}$$

Teoria dos Números

- Aritmética Modular: Basicamente, $a \equiv b \,(\mathrm{mod}\, n)$ se $a = b + kn$ para algum inteiro k. Se a for um não negativo e b estiver entre 0 e n, pode-se pensar em b como o resto de a quando dividido por n. b também é chamado de resíduo de a modulo n.

- Reversão Modular: executar a função modular inversa $a^{-1} \equiv x(\,\mathrm{mod}\, n)$ é uma tarefa difícil, podendo haver ou não uma resposta. Se a e n forem relativamente primos, existirá apenas uma solução para a^{-1}. Caso contrário, não existirão soluções. Se n é primo, então todo número de 1 a 1-n é relativamente primo e tem exatamente um inverso do $(\mathrm{mod}\, n)$.
  Existem métodos matemáticos para realizar esta inversão (Função Totient de Euler, Fatoração chinesa, etc.) mas a dificuldade de se implementar computacionalmente esta reversão para grandes valores de n e a é explorada nos sistemas de segurança.

- Geradores: Se p é um primo, e g é menor que p, então g é um GERADOR $(\mathrm{mod}\, p)$ se para cada b de 1 a p-1, existe algum a para o qual $g^a \equiv b \,(\mathrm{mod} p)$. Outra maneira de expor é g é primitivo em relação a p.

Criptografia por Substituição/Transposição e XOR

Algoritmos de criptografia implementam a segurança por meio de substituição ou transposição de dados.

- Substituição: neste tipo de criptografia, cada letra é deslocada por um substituto fixo, geralmente outra letra. A mensagem

$$M = m_1 m_2 m_3 m_4 ...$$

onde $m_1 m_2 ...$ são letras sucessivas se torna:

$$E = e_1 e_2 e_3 ... = f(m_1) f(m_2) f(m_3) f(m_4) ...$$

onde a função $f(m)$ tem uma inversa.

- Transposição: a mensagem é dividida em grupos de tamanho d e uma permutação aplicada ao primeiro grupo, a mesma permutação ao segundo grupo e assim sucessivamente. A permutação é a chave e pode ser representada pela permutação dos primeiro d inteiros. Exemplo: para d=5, pode-se ter como permutação 23154. O que significa:

$$m_1 m_2 m_3 m_4 m_5 m_6 m_7 m_8 m_9 m_{10} ...$$

se transforma em

$$m_2 m_3 m_1 m_5 m_4 m_7 m_8 m_6 m_{10} m_9 ...$$

Um exemplo prático é o cifrador de Vigenère, com comprimento d. Considerando um total de 26 letras sendo A=0 e Z=25, a função criptografia é:

$$e_i = m_i + K_i (\mathrm{mod} 26)$$

sendo K_i o período d e índice i.

- Função XOR: função lógica que implementa o cifrador de Vegenère, muito utilizada na implementação de algoritmos.

$0 \oplus 0 = 0$
$0 \oplus 1 = 1$
$1 \oplus 0 = 1$
$1 \oplus 1 = 0$

Figura 2.14: Tabela Lógica XOR

Tipos de Algoritmo de criptografia simétrica

- Blocos: divide o conteúdo em blocos de bits e criptografa o bloco todo de uma única vez.
- Cadeia [7] a criptografia é obtida para bit individualmente, adicionando 1 bit da cadeia individualmente.

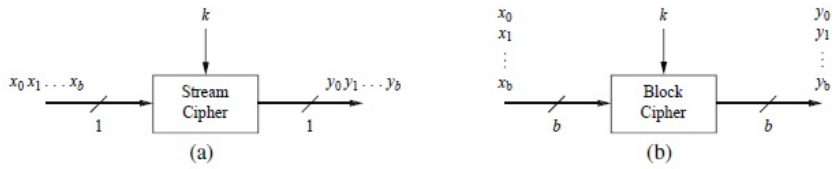


Figura 2.15: PAAR e PELZL [3](a) Criptografia de *Cadeia* (b) Em blocos

2.2.8 Algoritmos e Protocolos

- DES: algoritmo simétrico utiliza blocos de 64 bits e chaves de 56 bits. Homologado em 1992 como solução comercial, teve a morte decretada em 1998.
- AES/3DES: após a experiência com DES, em 1999 foi criado um padrão de segurança que especificava tamanhos padronizados de chaves e novos requisitos para segurança em comércio eletrônico. Vários algoritmos se candidataram para homologação. O DES com rodada tripla de criptografia foi utilizado temporariamente até que novos algoritmos estivessem disponíveis PAAR e PELZL [3]. Em 2000 o algoritmo de blocos *Rijndael* foi homologado.
- RC4: criado em 1987 por Ron Rivest Data Security, este algoritmo de cadeia foi proprietário por 8 anos mas em seguida passou a ser um padrão comercial *de fato* adotado pelos principais fabricantes no mundo. RSA desenvolveu, posteriormente o RC5 em blocos, que veio a se tornar uma família de algoritmos SCHNEIER [19] com diversos tamanhos de chaves e blocos.
- RSA: o nome é homenagem a seus criadores Ron Rivest, Adi Shamir, and Leonard Adleman RIVEST *et al.* [44]. Os detalhes de funcionamento serão apresentados adiante, no item 2.2.11. Este algoritmo trabalha com chaves assimétricas e tem grande utilização comercial, sendo utilizado para autenticação e assinatura digital principalmente.
- Curvas Elípticas: relativamente novas as implementações, mas os algoritmos de chave pública que utilizam curvas elípticas existem desde final dos anos 90.

[7] Tradução para *Stream Ciphers*

Sua performance em criptografia assimétrica é interessante porque, enquanto algoritmos convencionais (ex. RSA) utilizam grandes números como chaves (1024 a 3072 bits), curvas elípticas conseguem segurança similar com apenas 160-256 bits PAAR e PELZL [3].

- HASH e Assinatura Digital: algoritmos de hash atuam como "assinatura digital"de mensagens, gerando um padrão numérico de tamanho fixo (64 a 160 bits) que só pode ser associado a sua mensagem original M. A probabilidade de que qualquer outra mensagem gere um padrão semelhante é inviável computacionalmente. Os principais algoritmos disponíveis são MD4/MD5 e SHA. A associação com assinatura digital vem do seguinte:

  1- a criptografia com chave pública pode ser lenta e onerosa em termos de tamanho da saída. Logo, não é recomendável que seja utilizada para criptografar toda a mensagem.

  2- se fosse utilizada uma função hash para gerar um padrão numérico associado ao texto resultado e, em seguida, utilizada a chave pública da comunicação em questão para criptografar o resultado do hash (resultado é pequeno e pode ser criptografado sem problemas), então o dono da chave privada poderia atestar que realmente o texto que chegou é autêntico pois ele poderia utilizar a mesma função hash no mesmo texto enviado.

- TESLA (Timed Efficient Stream Loss-Tolerant Authentication): é um algoritmo para autenticação de mensagens tipo broadcast em comunicações móveis. Ele funciona como hash, gerando um padrão de saída único para a mensagem, que pode ser utilizada para autenticação do pacote CAYIRCI e RONG [5].

- WEP/WPA EDNEY e ARBAUGH [45] Introduzido em 1999 - WEP (Wired Equivalent Privacy), utiliza chaves de 40 a 104 bits e RC4 para criptografia. Sua segurança foi quebrada em diversos trabalhos GUPTA *et al.* [46],STUBBLEFIELD *et al.* [47], inclusive, tendo o RC4 sido apontado como o ponto frágil, o que rapidamente motivou sua defesa e erro de interpretação FLUHRER *et al.* [48]. O WEP foi substituído pelo WPA/WPA2 que utilizam AES 128 bits. Tanto o WPA quanto o WPA2 permitem o uso de servidores de autenticação (RADIUS), IEEE 802.1x e EAP - Extensible Authentication Protocol (RFC 2284/RFC 3748) (IEEE 802.1X, 2008).

- X.509 (Certificação Digital): o padrão ITU X.509 (RFC 2459) é utilizado para gestão de certificados digitais entre a Autoridade Certificadora (CA) e o elemento usuário do certificado YI *et al.* [49]. Quando uma implementação de

segurança necessitar de um estrutura de chaves públicas (PKI), este protocolo será utilizado para criação, emissão e revocação de certificados KENT e POLK [50].

2.2.9 Colisões em Algoritmos de HASH

Algoritmos de hash devem ser testados com ataques de colisão e aniversário PAAR e PELZL [3]. O assunto será melhor tratado matematicamente no apêndice B. Por ora, o assunto colisão em algoritmos de hash apenas será tratado em linhas gerais. Seja H um algoritmo de *hash* definido em $\{0,1\}^{\ell}$, sendo ℓ o tamanho em bits da saída gerada pela função. Suponha que sejam escolhidos q entradas distintas $x_1, ..., x_q$, gerando as seguintes saídas correspondentes: $y_1, ..., y_q$, gerados a partir de $y_i = H(x_i)$. A pergunta que se segue é: existiriam duas ou mais saídas y_i com valores iguais, gerando assim uma colisão? Ou, em um modo mais formal: Se forem escolhidos valores $y_1, ..., y_q \epsilon \{0,1\}^{\ell}$ de forma uniformemente aleatória, qual a probabilidade que existam valores iguais $y_i = y_j$, para $i \neq j$?

Este problema já foi amplamente abordado na literatura e também conhecido como Problema do Aniversário. No apêndice B é dada um solução genérica para este problema de colisão, aplicável para todos os cenários (aniversários, valores de *hash*, problemas clássicos envolvendo urnas e retirada, etc.). De uma perspectiva concreta de segurança, entende-se que um algoritmo que execute função hash deve resistir a um ataque de colisão que seja executado pelo tempo T (onde considera-se como referência de tempo a geração de valores de saída pelo algoritmo H), a saída do algoritmo deve ter como quantidade mínima de bits o valor $2 \log T$ bits (uma vez que $2^{(2 \log T)/2} = T$).

2.2.10 Processos para Gestão de Segurança

Entretanto, como visto em SCHNEIER [41] a segurança não está relacionada somente aos aspectos tecnológicos da solução, mas sobretudo ao comportamento. Aspectos fundamentais para a construção de um mecanismo de segurança:

- Segurança deve ser vista como um processo, não apenas um pacote de soluções;
- Nos processos, devem constar mecanismos para detecção, reação e isolamento de acessos maliciosos.
- Soluções complexas são avaliadas como menos eficientes;
- A motivação para implantação de soluções seguras encontra maior aderência em corporações que consigam identificar a rentabilidade em segurança ? ex.:

renovações de seguros.

Um aspecto crítico é a definição de planos de desastres, para casos de invasão, ataques de negação de serviço ou quebra de confidencialidade. A exemplo de desastres naturais, as ações devem ser imediatas e não há espaços para improvisos.
A garantia de sucesso em um plano de segurança está mais associada à disciplina de auditorias contínuas do que somente na técnica de criptografia empregada. As pessoas mudam, com elas os processos e tudo isso fará falta quando se estiver diante de uma ameaça real.

2.2.11 Geração e Troca de Chaves Públicas

- Diffie-Helmman (D-H): Diffie-Helmman DIFFIE e HELLMAN [51] apresentou a forma, ainda hoje mais utilizada para geração de chaves públicas e troca de chaves. Considerando-se Alice (origem) e Bob (destino), o processo D-H começa com Alice e Bob escolhendo um n primo grande e um g, desde que g seja primitivo de n (como visto em 2.2.7). Então:

  1- Alice escolhe um inteiro aleatório grande x e envia para Bob: $X = g^x \bmod n$ (chave pública da Alice);

  2- Bob escolhe um inteiro aleatório grande y e envia para Alice: $Y = g^y \bmod n$ (chave pública do Bob);

  3- Alice calcula: $K = Y^x \bmod n$ (chave privada da Alice);

  4- Bob calcula: $K' = X^y \bmod n$ (chave privada do Bob).

- Shamir: Shamir SHAMIR [52] publicou seu trabalho em 1979 demonstrando como gerar e compartilhar senhas com base na dificuldade de fatoração de grandes. As etapas são:

  1- escolher dois primos, p e q, de grande e igual tamanho: $n = pq$;

  2- escolha aleatoriamente o valor, e, tal que e and $(p-1)(q-1)$ sejam primos relativos. Finalmente, utilize o algoritmo estendido Euclidiano para calcular a chave de descriptografia, d:

  $$ed \equiv 1 \bmod (p-1)(q-1)$$

  ou ainda

  $$d = e^{-1} \bmod ((p-1)(q-1))$$

  3- d e n são primos relativos. Os números e e n são as chaves públicas; d é a chave privada. p e q não são mais necessários;

4- para criptografar uma mensagem m divididas em blocos m_i, executar $c_i = m_i^e \bmod n$.

5- para descriptografar, $m_i = c_i^d \bmod n$.

2.2.12 O Padrão Estabelecido Para Segurança Em Transações Eletrônicas: EMV

Uma série de padrões e especificações de como implementar transações seguras para ações de pagamentos foi especificada pelo grupo das principais fornecedoras de serviços de crédito no mundo. Este padrão é conhecido como EMV EMVCO [7] e tem orientações para implementação de serviços para cartões inteligentes, leitoras de chips em cartões de crédito, métodos para armazenamento seguro e formas de transmissão de segurança em redes de captura.

2.2.13 Falha no padrão EMV

Um grupo de pesquisadores de Cambridge BOND *et al.* [26] demonstrou que o padrão EMV com utilização de Chip e PIN de autenticação dos usuários pode ser interceptado a partir de falsificação dos leitores de cartões. Uma vez que algum dispositivo de leitura seja inserido no meio, é possível adquirir informações relevantes sobre a identidade do usuário, como nome do portador do cartão, número, código de segurança dentre outras informações críticas.

De posse destas informações, é possível por exemplo que se realize compras "on line"em sites de varejo, que normalmente solicitam informações apenas do cartão para a liberação da transação.

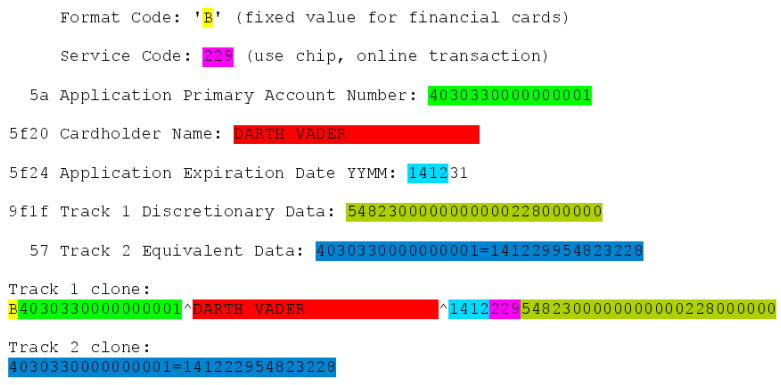


Figura 2.16: Vazamento de informação a partir da violação das máquinas leitoras de cartão

A técnica utilizada para o roubo de informações a partir de leitoras de cartão modificadas pelo grupo de Cambridge é conhecida como *Skimming*. Ela consiste em

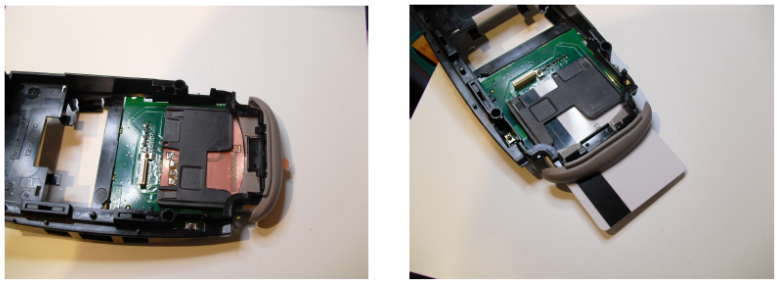


Figura 2.17: Processo de violação de leitores POS

violar fisicamente os leitores de cartão, substituindo partes críticas do equipamento por similares que irão executar os passos seguintes:

- Leitura do cartão: seja tarja magnética ou o chip de segurança. O processo de leitura é similar ao projeto original do leitor alterado, de modo que o usuário não consiga identificar qualquer diferença nos processos.
- Interceptação de dados: os equipamentos POS/TAD/ATM são desenhados para a leitura dos dados do cartão e o armazenamento temporário de alguns dados especialmente designados para a geração de sequências aleatórias, visando a autenticação de mensagens durante o processo de pagamento. Após a transação, estes dados são apagados da memória temporária no leitor. Entretanto, após um ataque tipo *Skimming*, esta memória temporária é alterada para o armazenamento destes dados que, posteriormente, serão acessados pelo criminoso que alterou o equipamento original. De posse destes dados, o criminoso poderá iniciar uma série de fraudes financeiras.

Na figura 2.16, estão representadas algumas informações críticas obtidas pelo método de violação tentado pelo grupo de Cambridge. Estas informações permitiriam a utilização do crédito do usuário "vítima"mesmo sem a clonagem da tarja magnética obtida facilmente também. Bastaria para tanto, as informações de identificação do usuário (o nome já seria suficiente) e as credenciais do cartão (número, data de expiração, código de segurança, etc. - todos contidos no cartão e facilmente adquiridos no processo de interceptação). Isso já possibilitaria a compra em qualquer varejista WEB, no qual uma falsa conta seja aberta associada as informações roubadas do cartão.

Na figura 2.17 pode-se ver fisicamente a ação de *Skimming*, que consiste na violação física dos leitores de cartão para a execução dos passos mencionados anteriormente. Estes equipamentos seguem normas de segurança rígidas estabelecidas para controlar a construção e homologação dos mesmos pela mesma entidade EMV que estabelece o controle dos processos. As normas estabelecem o nível de criptografia e

proteção sobre as informações armazenadas (temporariamente ou não) hardware envolvido. Entretanto, a evolução tecnológica das quadrilhas especializadas em crimes contra o sistema financeira é bastante facilitada pelo fato das informações estarem disponíveis facilmente para interessados nesta tecnologia.

2.2.14 Definição de Entropia e Aleatoriedade

A Entropia no contexto deste trabalho é definida no trabalho de SHANNON [53], o qual estabelece a Entropia como uma medida da incerteza dos valores obtidos a partir de uma variável aleatória. Ou seja, em termos matemáticos, como a seguir.

Seja X uma variável aleatória discreta cujos valores estão contidos em χ e com função massa de probabilidade $p_X(x) = Pr(X = x)$, $x \epsilon \chi$. A Entropia $H(X)$ de uma variável aleatória discreta X é definida por

$$H(X) = -\sum_{x \epsilon \chi} p(x) \log p(x) \tag{2.9}$$

Se o logaritmo da equação 2.9 utilizar base 2, a entropia é medida em *bits*. Por exemplo, a entropia de uma moeda honesta sendo lançada em um experimento simples é de 1 *bit*. Se a base utilizada for e, a entropia é medida em *nats*. Uma vez que não seja explicitamente mencionada, a base será sempre 2 e, ao longo do texto desta dissertação, a entropia sempre considerada *bits*.

Considerando duas variáveis aleatórias X e Y (cujo conjunto de valores está contido em γ, de modo que $y \epsilon \gamma$), com respectivas funções massa de probabilidade definidas por $p_X(x)$ e $p_Y(y)$, outros dois conceitos importantes são a **Entropia conjunta** $H(X, Y)$ (considerando a distribuição conjunta $p(x, y)$ como sendo existente)

$$H(X, Y) = -\sum_{x \epsilon \chi} \sum_{y \epsilon \gamma} p(x, y) \log p(x, y) \tag{2.10}$$

e a **Entropia condicional** $H(Y|X)$ (considerando $(X, Y) \approx p(x, y)$) como sendo

$$H(Y|X) = \sum_{x \epsilon \chi} p(x) H(Y|X = x) \tag{2.11}$$

$$= -\sum_{x \epsilon \chi} p(x) \sum_{y \epsilon \gamma} p(y|x) \log p(y|x) \tag{2.12}$$

$$= -\sum_{x \epsilon \chi} \sum_{y \epsilon \gamma} p(x, y) \log p(y|x) \tag{2.13}$$

Conforme já mencionado no item 2.2.7, o conceito de entropia está diretamente associado à capacidade de se estabelecer níveis de segurança adequados, de acordo com a criticidade de uma aplicação. Uma vez especificado que o método para proteção das informações críticas é a criptografia das mesmas, então a preocupação com a entropia se faz necessária. Os algoritmos criptográficos necessitam de fontes de entropia que sejam próximas o suficiente da aleatoriedade teórica (encontrada apenas na natureza) para conseguirem gerar números aleatórios criptograficamente seguros. Existe uma recomendação do IETF (*Internet Engineering Task Force*), que estabelece as regras de interconexão de dispositivos e serviços em forma de recomendações (amplamente adotadas em normas técnicas como ISO/IEC), para que estas fontes de entropia sigam um padrão de funcionamento e mantenham-se protegidas dentro do projeto em questão. Esta recomendação, RFC (designação IETF para suas recomendações) 4086 SCHILLER e CROCKER [54] cujo capítulo para as fontes sugeridas de entropia indica as seguintes origens:

1. Discos existentes.
2. Osciladores de relógio.
3. Placas de som e vídeo.
4. Medição de tempo entre eventos externos na natureza.

Quanto a **aleatoriedade**,

2.2.15 Principais Métodos Para Geração de Números Aleatórios

Geração de pseudo-Aleatoriedade

Sistemas de segurança que implementam criptografia necessitam gerar sequências aleatórias para executarem a substituição de caracteres ou embaralhamento dos mesmos. Entretanto, os algoritmos conseguem apenas se aproximar de uma aleatoriedade real, somente encontrada na natureza. Desta forma, determina-se como pseudo-aleatoriedade a capacidade de funções matemáticas proverem sequências binárias com a entropia (tal qual visto em 2.2.14) mínima necessária. A utilização de algoritmos ineficientes para a geração de entropia mínima, colocam em risco a segurança de todo o sistema por deixar o processo previsível suficiente para que um ataque de força bruta (capaz de implementar tentativas infinitas a fim de prever valores e comprometer a capacidade do sistema de implementar o conceito "perfeitamente seguro"discutido em 2.2.7 (ainda que hipoteticamente falando, uma

vez que qualquer implementação prática esteja fora da definição de perfeita).

Grande esforço foi empregado na construção de geradores pseudo-aleatórios e ampla literatura está disponível. No que diz respeito a criptografia, nem todo gerador atende os critérios que um sistema criptográfico exige, como alta entropia e robustez contra ataques diversos que tentem prever valores gerados. Algoritmos que implementem estas duas características são conhecidos como **criptograficamente seguros** (SCHNEIER [19]). Geradores também são amplamente utilizados para simulações diversas, tais como tráfico de redes, veículos, mobilidade humana, etc. Para estes geradores, não existe o rigor para que atendam critérios criptográficos. A seguir, os principais geradores

Tipos de Geradores de Números Pseudo-Aleatórios

Geradores de números aleatórios são funções matemáticas capazes de fornecerem sequências finitas de números, com probabilidade de repetição baixa o suficiente para serem consideradas improváveis. Além disso,

- São algoritmos que necessitam de um valor inicial para início do processo de geração de números. Este valor inicial (chamado vetor de inicialização) é composto por um valor inicial conhecido como **semente** e um valor arbitrário que represente o estado inicial do algoritmo. A boa prática da implementação destes geradores estabelece que estes valores iniciais devem ser obtidos por meio de uma fonte de entropia real, ou seja, a partir de algum fenômeno natural do qual seja retirada a entropia real para alimentar o sistema.
- São determinísticos: ou seja, quando a entrada e as condições de inicialização são repetidas, o mesmo valor é gerado.
- São finitos, mas com grande período de repetição.
- O valor a ser gerado depende do valor gerado anteriormente, ou seja, $Y_{i+1} = f(Y_i)$, onde Y_i é saída gerada pela função f no momento i.
- Geradores de Números Pseudo-Aleatórios criptograficamente seguros possuem uma propriedade que os distinguem dos demais: não previsíveis. Ou, matematicamente definindo segundo PAAR e PELZL [3], dadas n sequências binárias Y_i consecutivas geradas, não há nenhum algoritmo de tempo polinomial que permita prever com ao menos 50% de chance o próximo valor de saída, Y_{n+1}.

Os PRNGs[8] mais conhecidos são apresentados a seguir:

[8] do inglês: Pseudo Random Number Generators - será utilizada esta forma para designação dos algoritmos para geração de pseudo-aleatoriedade.

- Geradores Congruentes Lineares: são geradores amplamente divulgados e possuem a seguinte fórmula de recorrência:

  $X_n = (aX_{n-1} + b) \bmod m$ nos quais X_n é o n-ésimo número da sequência e X_{n-1} é o valor prévio da mesma sequência. As varáveis a, b e m são constantes: a é o multiplicador, b é o incremento e m o módulo. A chave ou semente, é o valor de X_0. Este gerador possui período menor ou igual a m. Se os parâmetros forem escolhidos adequadamente, este gerador alcançará o período máximo de geração e o período será igual a m. Estes não são geradores criptograficamente seguros, pois são previsíveis.

- Geradores de Registradores de Deslocamento com Realimentação Linear

  São compostos por registradores de bits conhecidos por sua função lógica por "*flip-flop*". Os estados dos registradores podem ser modeladas em polinômios cuja ordem é a mesma quantidade de registradores. Esta estrutura também é utilizada para geração de sequências cifradas. Existem vulnerabilidades conhecidas para as versões lineares deste tipo de gerador, de modo que seu uso fica restrito quando se necessita de geradores criptograficamente seguros. Um exemplo de construção é demonstrado na figura 2.18.

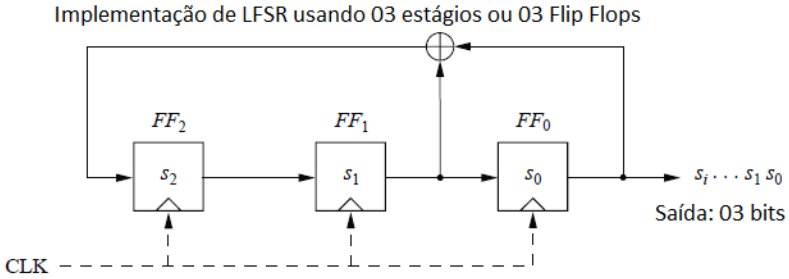


Diagrama de estados dos Flip Flops

clk	FF_2	FF_1	$FF_0 = s_i$
0	1	0	0
1	0	1	0
2	1	0	1
3	1	1	0
4	1	1	1
5	0	1	1
6	0	0	1
7	1	0	0
8	0	1	0

Figura 2.18: Construção de um gerador utilizando 03 *Flip Flops* como registradores linearmente conectados.

2.2.16 Métodos Para Avaliação da Aleatoriedade

A melhor referência para medições diversas e avaliação da aleatoriedade é NIST (National Institute of Standards and Technology) que disponibiliza tanto o método

como um código para a implementação em software destes métodos NIST [55]. Este método será adotado nesta dissertação.

2.2.17 Fortuna

O Fortuna (FERGUSON e SCHNEIER [20]) é um método para geração de números pseudo-aleatórios que inovou a forma de obtenção destes números para alimentar algoritmos de criptografia. Ele implementa algumas melhorias em relação a sua versão anterior, o *Yarrow*, principalmente na construção de um acumulador de entropia e também no melhor gerenciamento das "sementes"necessárias para seu funcionamento. A figura 2.19 ilustra sua estrutura.

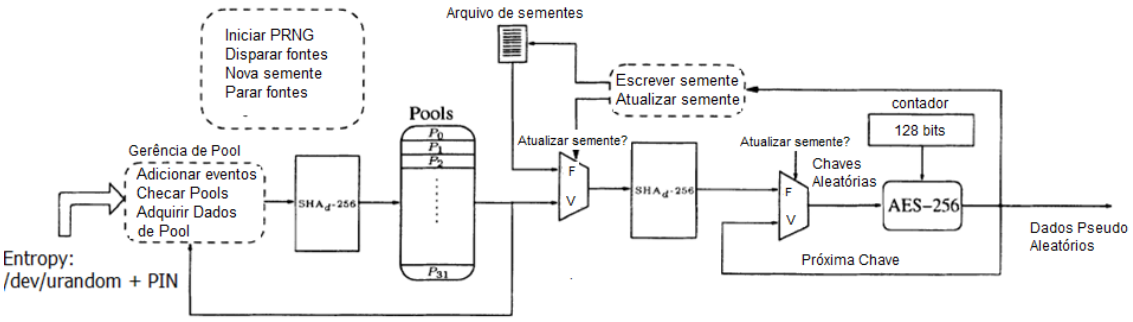


Figura 2.19: Fortuna: Método para geração de números pseudo-aleatórios.

O Fortuna é constituído por três partes principais: um Gerador de Números, um Acumulador de Entropia e um Gerenciador de Sementes.

Gerador de Números

O Fortuna utiliza um gerador que implementa o algoritmo AES em modo de contagem (CTR), com qualquer implementação dentre as implementações mais conhecidas (Rijndael, Serpent, ou Twofish). O estado interno é constituído por 256 bits, com contador binário de 128. Cada etapa de criptografia é feita em blocos de 128 bits. Se fossem permitidos a entrega de 2^{64} blocos a cada requisição, é esperado que se tenha ao menos um valor repetido. Para diminuir a probabilidade de colisões 2.2.9, o gerador é limitado a apenas entregar 2^{16} blocos a cada solicitação, limitando sua saída a $1MB$. Se um usuário necessitar de mais valores, então a solicitação é feita novamente. Sendo um algoritmo determinístico (uma das características fundamentais de um PRNG), está sujeito a ataques que comprometam o estado interno (basicamente, quando um atacante demanda uma quantidade estressante de números aleatórios de modo que o gerador não consiga atualizar seu estado interno, renovando a semente inicial e partindo de novo vetor de inicialização).Para se proteger deste tipo de ataque, a cada nova solicitação de números o gerador utiliza dois blocos (256 bits) para

renovar sua chave interna. A antiga chave é descartada. Com o limite de 2^{16} blocos a cada solicitação, a probabilidade de colisão é de aproximadamente $\frac{(2^{16})^2}{2.2^{128}} = 2^{-97}$. Isso significa que um atacante deverá fazer 2^{97} requisições de 2^{16} blocos com a mesma chave, o que já foi posto não ser possível devido a renovação de chave a cada nova solicitação.

Acumulador de Entropia

O Fortuna utiliza até 256 fontes de entropia real (interrupções de CPU, oscilações de discos, etc.), das quais são retiradas as sequências aleatórias que irão possibilitar a gerar os valores de semente e chave. São concebidas filas de armazenamento desta entropia, que podem ser designadas pelo termo original inglês *Pool* na bibliografia disponível. No total, 32 filas de 256 bits cada. No total, pode armazenar até $32.256bits = 8192bits$ de entropia. Antes de serem armazenadas , a saída de cada fonte de entropia é direcionada para uma função hash de SHA256, garantindo que a entrada para cada fila será a mais próxima possível da entropia máxima no sistema.

Gerenciador de Sementes

Uma função *hash* de SHA256 também é executada para garantir uma atualização de valores da chave serão feitas a partir de valores de novas sementes e da chave anterior, concatenadas matematicamente. Além disso, durante o funcionamento do gerador, um arquivo contendo valores de sementes é atualizado de modo que se tenha valores adequados e disponíveis para cada funcionamento do gerador após uma etapa de reinicialização de todo o sistema, ficando assim o gerador menos vulnerável a qualquer deficiência do sistema em fornecer fontes de entropia criptograficamente adequadas.

2.3 Gestão de Identidade

Um dos pontos mais importantes deste trabalho, é a tese de que nos meios tradicionais de pagamentos a identidade do usuário (suas credenciais de segurança, identificação junto ao provedor de serviços financeiros, etc.) está, por diversas vezes exposta, devido ‘a evolução das novas técnicas de interceptação do processo de pagamento, tal como fora visto na seção 2.2.13. Uma inspiração para esta dissertação foi o trabalho apresentado em PUJOLLE *et al.* [6] sobre a *SecFuNet* (projeto coordenado pelo CNPq), no qual a gestão da identidade do usuário é feita por elementos virtuais disponíveis em infraestrutura de alta disponibilidade (*cloud computer*), conforme ilustrado na figura 2.20.

Figura 2.20: Figura baseada em PUJOLLE *et al.* [6] descrevendo a gestão de identidade feita para o projeto SecFuNet.

A identidade do usuário é "portada"através de um cartão inteligente ("smartcard"), normalmente que utiliza tecnologia *JAVA CARD* para desenvolvimento de aplicações (*applets*). Uma vez inserido em algum leitor compatível, este cartão é alimentado eletricamente e consegue executar os *applets*, podendo interagir com aplicações externas fornecendo informações como, por exemplo, credenciais de segurança de um determinado usuário. Este processo acontece com transações que utilizam cartões de crédito e autenticação de usuários baseada em PIN. O elemento de segurança de um cartão de crédito (o "chip"de segurança do cartão) controla todo os processo de autenticação/autorização da transação, passando para a máquina leitora (o POS conhecido vulgarmente como "maquininha"de pagamentos) as credenciais para criptografia da comunicação, sequências numéricas para assinatura da transação, dentre outros dados.

Os cartões com a identificação de cada usuário são mantidos pelo sistema centralizado de gestão, através de comunicação segura TLS (*Transport Layer Security*). Este protocolo é definido pela RFC 5746. É utilizado para estabelecimento de comunicação segura entre dois agentes, implementando criptografia pública assimétrica (ver seção 2.2.6) e certificados digitais padrão X.509[9]. Estes certificados podem ser instalados diretamente nos cartões de identificação dos usuários, sendo aplicados como fontes com informações de credenciais para estabelecimento de comunicação segura.

Por fim, é importante destacar que a plataforma Android possui interfaces com diferentes modalidades de Elementos de Segurança (virtuais ou baseados em hardware), podendo ser utilizada para implementar este mesmo tipo de autenticação

[9]Padrão ITU-T para infra-estrutura de chaves públicas.

remota sugerida por PUJOLLE *et al.* [6].

2.4 Linux e Android

2.4.1 Entropia e Geração de Números Aleatórios no Linux e Android

Segundo GUTTERMAN *et al.* [16] o Android herdou a estrutura de geração de números aleatórios do Linux. A estrutura é representada na figura 2.21.

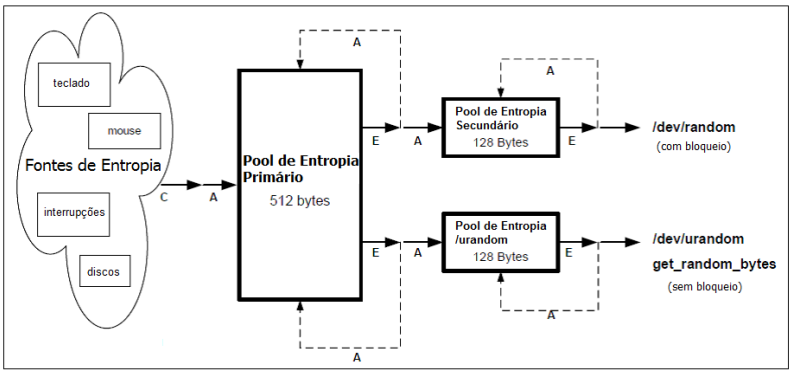


Figura 2.21: Estrutura de geração de números aleatórios do Linux .

As principais características deste modelo são:

- Fonte de entropia fornecida por hardware e armazenada em filas de entropia (512 bytes de números aleatórios "reais");
- Filas específicas para alimentação de processos e aplicações internas, com distintas características: /dev/random armazena 128 bytes fila original original, bloqueando o fornecimento para valores menores que 92 bytes [10], e /dev/urandom que desempenha função de PRNG (portanto, com capacidade de fornecimento de entropia sem necessidade de bloqueio);
- A estrutura funciona como o modelo Fortuna originalmente concebido por FERGUSON e SCHNEIER [20]. /dev/urandom utiliza os valores de /dev/random como "seed" e alteração do estado de forma a não ser estaticamente possível definir seu próximo estado. Entretanto, a execução apresenta problemas, o que será visto posteriormente.

A extração da entropia é um processo complexo e requer a utilização de um registro de deslocamento de bits TGFSR (*Twisted Generalized Feedback Shift Register*, de acordo com SCHNEIER [19]), com palavras de 32 bits.

Uma série $\alpha_i \in \{0,1\}^w$ e uma matriz $A_{w\times w}$

$ATGFSR$

$$\begin{aligned} & x^P + x^{P-1} + ... + x^{P-t_m} + 1(1 \leqslant t_{1,...,t_m<p}) \\ & if\ \alpha i = \alpha_{i-p+t_i} \oplus ... \oplus \alpha_{i-p+t_m} \oplus ... \oplus \alpha_{i-p}\mathrm{A} \\ & i = p, p+1, ... \end{aligned} \tag{2.14}$$

A equação 2.14 é o polinômio corresponde à saída do registrador. A extração de entropia, no total 32 bits disponíveis para utilização do gerador, é feita a partir da conversão dos fenômenos nos dispositivos de hardware, tais como ruídos elétricos, oscilações dos discos, interrupções da CPU, dentre outros, em bits para os acumuladores formados pelos registradores. O estado inicial dos registradores é definido unicamente pelo valor inicial ("seed") a cada momento de inicialização do gerador.

2.4.2 Fontes de Entropia no Linux e Android

Há três fontes de entropia: discos (Linux, não Android), interrupções e entropia inserida pela manipulação dos usuários. O Android ainda oferece algumas outras alternativas devido ao fato de possuir outros sensores embarcados, tais como giroscópios, GPS, acelerômetros.

Em ambos os sistemas, a forma de se extrair a entropia corrente é através da variável de ambiente

```
/proc/sys/kernel/random/entropy_avail
```

As definições de fontes de entropia localizam-se no arquivo `random.c`. Neste arquivo encontram-se todas as varáveis de ambiente e os processos que irão definir a entropia total do sistema. Abaixo, exemplo destas especificações.

```
#include <asm/irq_regs.h>
#add_disk_randomness(struct gendisk *disk)
#ifdef ADD_INTERRUPT_BENCH
#add_timer_randomness(&input_timer_state,
```

2.4.3 Falhas de Segurança do PRNG Linux e Android

Recentes trabalhos, têm reportado falhas que submetem o modelo acima descrito a uma nova avaliação de segurança. Falhas que não podem ser exploradas facilmente, mas que podem vir a comprometer um complexo e crítico sistema de pagamento

eletrônico.

No trabalho apresentado por DODIS *et al.* [17], são citados como principais fontes de problema:

1. O método de estimação da entropia: quando as fontes de entropia não conseguem, por algum fenômeno qualquer, fornecer a quantidade de bits esperada pelo gerador, o sistema empregado para compensação é atualizar o gerador com a chave anteriormente utilizada, sem a renovação do estado interno sendo feito por uma entrada de entropia real. Este método é condenado por aumentar a vulnerabilidade do gerador frente a ataques que utilizam "força bruta", levando o gerador a um nível de estresse e forçando uma situação de colisão.

2. A deficiência no armazenamento e obtenção de novas sementes (*seeds*, doravante para melhor aderência às fontes bibliográficas).

KAPLAN *et al.* [18] demonstrou que devido a baixa entropia do sistema imediatamente após o *startup* ou *boot*, o PRNG (*Pseudo Random Number Generator*) do Android permite um ataque conhecido como *Stack buffer Overflow* , no qual é possível a um aplicativo malicioso se apropriar de um espaço de memória de outra aplicação de modo a adquirir informações críticas. Devido a condição de baixa entropia, a defesa contra este tipo de ataque (chamada *stack canary*) perde significativamente sua eficiência e expõe todo o sistema.

DING *et al.* [8] apresenta um estudo comprovando a presença de entropia suficiente no sistema $\backslash dev \backslash random$ *e* $\backslash dev \backslash urandom$, desmistificando o fato de muitos desenvolvedores que se opunham. Mas, ao mesmo tempo, confirmou o trabalho anterior reafirmando a necessidade de melhorias na entropia após o processo de *boot* de todo o sistema, pelo mesmo motivo que KAPLAN *et al.* [18]: ataques de *Stack buffer Overflow*.

2.4.4 Descrição do Sistema Android

O Android foi inicialmente desenvolvido por uma empresa chamada Android Inc., em 2003 e adquirida pelo Google em 2005. É um sistema baseado em Linux que possui diversas camadas de sistemas, bibliotecas, etc. Existe uma camada sobre a qual todos os aplicativos desenvolvidos e instalados pelo usuário rodam, chamada *Dalvik Virtual Machine*. Esta aplicação é extremamente importante porque ela é sempre replicada para execução dos aplicativos baseados em Java.

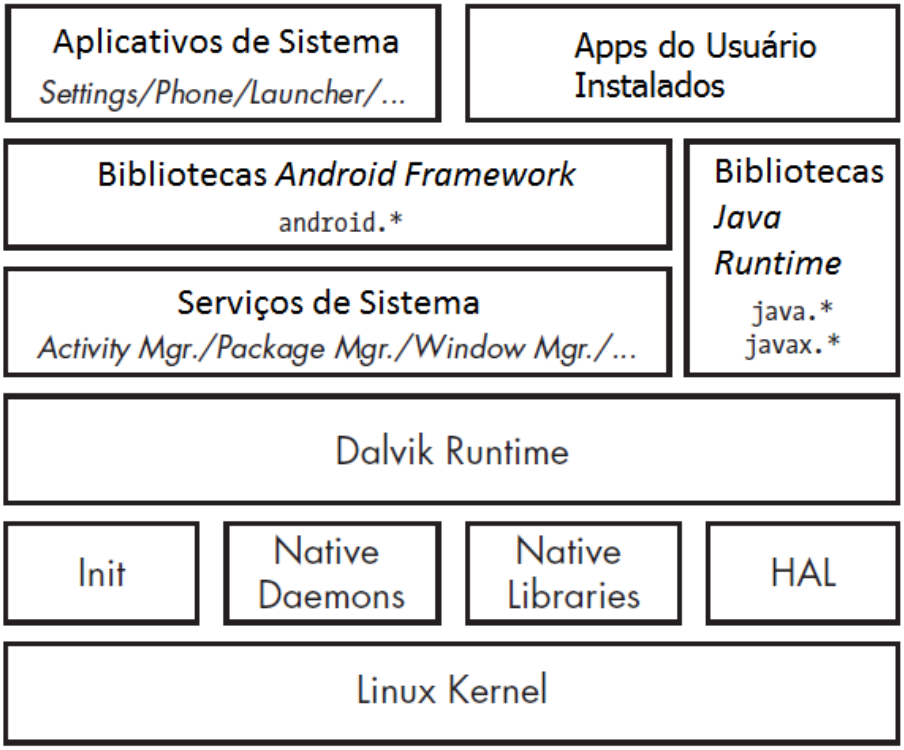


Figura 2.22: A arquitetura Android

Kernel Linux

Como pode ser visto na figura 2.23 (ELENKOV [28]) o Android é construído no topo do Kernel [10] Linux. Como qualquer sistema Unix [11] disponibiliza os controladores de hardware, redes, sistema de arquivos e gerenciamento dos processadores. Graças ao projeto "*Android Mainlining*"[12] o Kernel Android é ligeiramente diferente de um uma versão (distribuição) Linux convencional que pode ser encontrada disponível para micro computadores *desktop*. As diferenças recaem sobre um conjunto de novas funções (por vezes citadas na literatura como *Androidisms* [13] adicionadas para suportarem o Android. Alguns dos principais Androidisms são menor consumo e administração de memória, melhor administração de processos (entrar em modo de baixo consumo quando não solicitado), compartilhamento de memória com anonimato (*ashmem*), alarmes, novas ferramentas para redes inseguras (*paranoid networking*) e compartilhamento de ambientes e recurso, conhecido pelo termo *Binder*. *Binder* implementa o IPC (Comunicação Inter Processos), que será visto posteriormente.

[10] Kernel: refere-se a núcleo do sistema operacional, onde se encontram as funções administrativas mais restritas. Termo normalmente não traduzido na literatura.

[11] Sistema operacional desenvolvido por Ken Thompson e Dennis Ritchie em 1971, Laboratórios Bell - fonte: *www.unix.org*

[12] *Android Mainlining Project*, *www.elinux.org*

[13] Para maiores detalhes sobre *Androidisms*, ver em Embedded Android, Karim Yaghmours, O'Reilly, 2013, pp. 29-38

Espaço de Usuário Nativo

Logo acima do Kernel, está a camada de espaço do usuário, consistindo dos aquivos *init* (o primeiro processo iniciado na partida do sistema), vários programas e sub-rotinas independentes que são iniciadas pelo arquivo *init* e executados em plano de fundo conhecidos como *deamons*, além de inúmeras bibliotecas utilizadas pelo sistema. Embora esta seja uma estrutura legada dos sistemas baseados em UNIX, as implementações Android possuem características funcionais próprias, diferindo das originais Linux no detalhe do seu desenvolvimento.

Dalvik VM

A máquina virtual Dalvik (*Dalvik Virtual Machine* - DVM) é o interpretador padrão Java [14] para as aplicações Android que utilizam o Java como linguagem nativo em seu desenvolvimento. Uma máquina virtual Java (*Java Virtual Machine - JVM*) da SUN é um desenvolvimento feito em pilhas, devido ao fato de conseguir ser executada em qualquer hardware. A implementação JVM foi feita tendo em mente dispositivos móveis e não pode executar códigos java diretamente (arquivos *.class*): seu formato de arquivo padrão nativo é chamado *Dalvik Executable (DEX)*, cujos arquivos possuem a extensão *.dex*. Desta forma, arquivos *.dex* são armazenados tanto na biblioteca Java(arquivos JAR) quanto junto as aplicações Android (arquivos APK).
A máquina virtual Dalvik (DVM) e JVM da Oracle têm diferentes arquiteturas - e diferentes conjuntos de instruções. Devido a performance, a DVM é iniciada apenas uma única vez após a partida do sistema. Cada nova aplicação iniciada no Android, copia as características da DVM e cria uma instância independente. Este processo de replicar a DVM em várias instâncias diferentes é feito por um processo chamado *Zygote*.

Inicialmente, o processo *Zygote* carrega as principais classes Android, mantendo-as no topo da lista de memória utilizada. Em seguida, ele monitora se novos comandos para iniciar uma nova aplicação Android serão disparados pelo sistema. Ao receber tal comando para alguma aplicação, ele executa um processo chamado *fork* [15], criando um novo processo e carregando a aplicação solicitada pelo sistema, pré-inicializando a mesma com as classes Android anteriormente carregadas em memória na etapa anterior.

[14] Linguagem de programação pertecente e mantida por Sun Microsystem/Oracle
[15] Fork: função nativa de sistemas UNIX, através da qual um processo replica a si mesmo, criando uma outra instância.

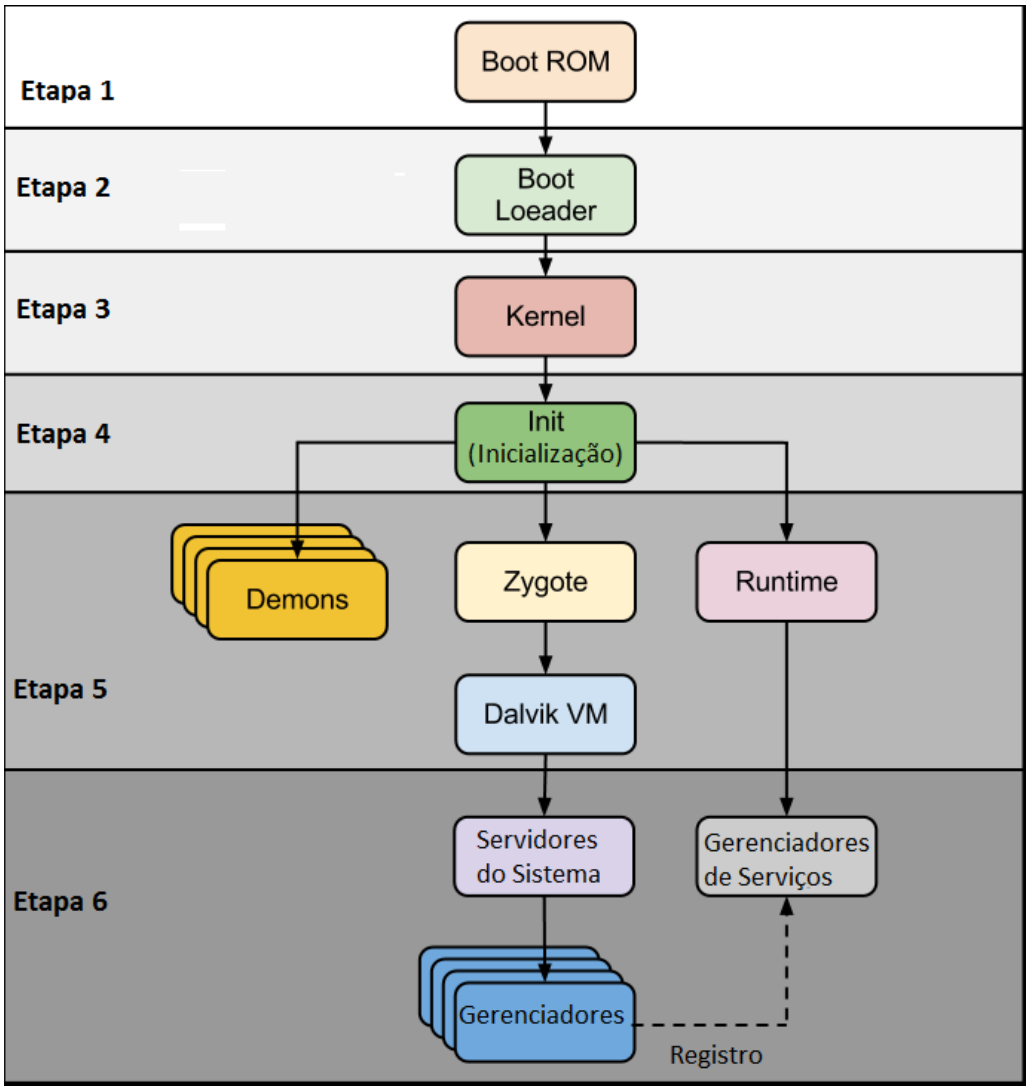


Figura 2.23: Passo a passo as etapas da inicialização Android, destacando a criação do *Zygote* e camada Dalvik.

Biblioteca Java

Uma implementação de linguagem Java requer pacotes de bibliotecas. O núcleo de Java do Android foi originalmente a partir de um projeto chamado "*Apache Harmony*"[16] e está na próxima camada de referência. À medida que o Android se desenvolvia, o código original *Harmony* foi sendo alterado significativamente. No processo, algumas características foram substituídas inteiramente (como suporte internacional, provedor de criptografia dentre outras hoje relacionadas as classes Java) enquanto outras foram estendidas e melhoradas. O código nativo Android é associado a biblioteca Java através do padrão *Java Native Interface (JNI)*[17], a qual permite o código Java chamar códigos nativos e vice versa. Esta camada é acessada diretamente pelos serviços do sistema e pelas aplicações.

[16] *The Apache Software Foundation, Apache Harmony, www.harmony.apache.org.*
[17] Oracle, Java Native Interface, www.docs.oracle.javase

Serviços do Sistema

As camadas introduzidas até agora, fizeram as adaptações necessárias para o núcleo Android (baseado em Linux) funcionar: A camada de Serviços do Sistema. Estes serviços de sistema (79 serviços documentados até a versão 4.4 do Android) implementam a maioria das características fundamentais do Android, incluindo o "*touch screen support*", telefonia e conectividade de rede. A maioria destes serviços é implementada em Java - algumas são escritas em código nativo Linux.
Com algumas exceções, cada serviço de sistema define uma interface remota que pode ser chamada por outros serviços e aplicações. Nesta camada do sistema, algumas das mais importantes funções são executadas e fazem parte das funções críticas do Android: Comunicação Inter-Processo (IPC[18]) e *Binder*.

Comunicação Inter-Processo

Como mencionado, *Binder* é um mecanismo de comunicação inter-Processo, IPC. Antes de detalhar o *Binder*, parte-se para uma descrição do IPC.
Como um sistema baseado em Unix, os processos no Android têm separados espaços de endereçamento e não podem acessar a memória alocada para outros processos - o que é chamado isolamento de processo. Isso é um procedimento de segurança, uma vez que múltiplos processos compartilhando memória poderia ser desastroso para o sistema, com um dos processos sobrecarregando mais a memória disponível que outro, por exemplo.Porém, se algum processo precisa fornecer alguma informação para outro, será necessário implementar alguma solução que o faça de forma segura sem comprometer a estabilidade do sistema como um todo.
Esta situação e a necessidade de tal mecanismo não é nova e data de muito tempo antes do Android. Isso inclui arquivos, sinais, filas, *sockets* TCP/IP, dentre outras funcionalidades.

Binder

Devido ao fato de que o IPC padronizado para sistemas baseados em Unix não era flexível e confiável o suficiente, um novo IPC chamado *Binder* foi desenvolvido para Android. Trata-se de uma nova implementação porém baseada na idéia de *OpenBinder*[19]. *Binder* implementa uma arquitetura baseada em interfaces abstratas, com componentes distribuídos. É similar ao *Windows Common Object Model* (COM) e *Common Object Broker Request Achitectures* (CORBA) no Unix, mas diferentemente destes, ele funciona em um aparelho único e não suporta chamadas remotas

[18] *Inter-Process Communication* - designação pela qual esta função será referenciada a partir de agora
[19] PalmSource, Inc., *OpenBinder*, *www.angryredplanet.com*

de processamento (RPC) através de rede (embora o suporte para RPC possa ser implementado como aplicação em camada acima do *Binder*. Uma descrição detalhada do funcionamento do *Binder* está fora do escopo desta dissertação, mas serão introduzidos alguns componentes principais brevemente, uma vez que o *Binder* é um elemento crítico no aspecto de segurança do Android.

- Implementação do *Binder*: implementado como um dispositivo virtual */dev/binder* no núcleo Android.
- Segurança do *Binder*: somente o processo de alta prioridade (*root*) pode executar uma chamada na função do *Binder*, de modo que qualquer aplicação que deseje utilizar este recurso deverá estar previamente autorizada.

2.4.5 Modelo de Segurança do Sistema Android

A base de segurança do sistema Android é o isolamento dos dados utilizados por uma aplicação. Devido ao fato do Android ter sido desenvolvido para plataformas *smartphones*, o que pressupõe uma utilização individual do aparelho, algumas características ficaram mais fragilmente implementada tais como: identificação física do usuário (a partir da versão 4.4 *Kit Kat* foram introduzidas bibliotecas associadas a leitura biométrica, mas somente em versões recentes de aparelhos) e a administração de privilégios do super usuário (em tese, qualquer aplicativo pode executar funções de administração, o que resulta em procedimentos facilmente executados pelo próprio usuário, criando o risco de todo o núcleo ser afetado proposital ou acidentalmente neste processo). As principais funções associadas a segurança do sistema são as seguintes:

- Isolamento de Aplicações: função conhecida como "*Sandboxing*", que identifica cada aplicativo com um número ID individual. Desta forma, cada espaço de memória (do aparelho, cartões externos) passa a ser monitorado pelo núcleo do sistema de modo que somente o ID permitido possa acessar e manipular os dados armazenados.
- Permissões: devido ao motivo acima exposto, os bloqueios a conteúdos permitidos somente pode ser liberado dentro da política de permissões, estabelecida para cada aplicativo no momento da sua instalação[20]. Na instalação do aplicativo, o sistema armazena as autorizações feitas pelo usuário em arquivo chamado *AndroidManifest.xml*. O Android inspeciona este arquivo toda vez que um aplicativo solicita acesso a algum recurso no sistema.

[20]Durante a instalação, é solicitado ao usuário a liberação ou não de acesso do aplicativo em questão a diversas informações disponíveis no sistema, tais como: fotos, dispositivos GPS, mensagens SMS, etc. Normalmente, os usuários ignoram a maioria destas solicitações e permite o acesso solicitado sem critérios ou preocupações com impactos futuros.

- IPC: como citado anteriormente no item sobre o *Binder*, o IPC (*Inter Process Communication*) é a função do núcleo Linux para uma aplicação ter acesso e controle sobre recursos de hardware ou software. Como exemplo, uma aplicação que deseje acessar algum dado externo por meio de *Bluetooth*[21], ele deve requerer autorização do usuário durante sua instalação, a qual será armazenada adequadamente no arquivo *AndroidManifest.xml*.

[21]Protocolo de comunicação de rádio frequência para curtas distâncias entre dispositivos eletrônicos.

Capítulo 3

Método Proposto

3.1 Método de Geração de Tokens

Como visto na seção 2.4.3, existem questionamentos sobre a robustez do processo de geração de números aleatórios dos sistemas Linux e Android (conforme já citado ao longo do trabalho, sistemas que estão historicamente correlacionados em seus fundamentos). A proposta desta dissertação não é propor melhoria nestes sistemas mas propor um método que aumente a segurança de transações eletrônicas a partir destes sistemas.

Desta forma, como evolução conceitual dos métodos de pagamentos eletrônicos, propõe-se a utilização de um método alternativo para proteção da identidade dos usuários do sistema financeiro, alterando a topologia de propagação de dados críticos tal como é feito hoje nas redes de captura de cartões de crédito, onde o PAN, *Primary Account Number* (ou efetivamente o número do cartão de crédito e suas credenciais de segurança) propagam na rede de captura e são normalmente armazenadas nos aplicativos que utilizam métodos de pagamento.

Este método é previsto no conjunto de normas especificadas para transações financeiras, já mencionada anteriormente chamado EMV EMVCO [7]. Conhecido como *Tokenization*, doravante mencionado como geração de tokens, ele consiste em substituir a credencial de identidade por uma sequência pseudo-aleatória de caracteres e que mantenham uma correspondência única com a identidade original. O emissor deste token será o único no processo a saber a correspondência exata entre identidade e o token correspondente. A figura 3.1 foi retirada do capítulo que trata da geração de tokens do processo.

A metodologia adotada nesta dissertação é baseada na utilização de soluções *opensource* para a construção de um modelo conceitual desta evolução tecnológica.

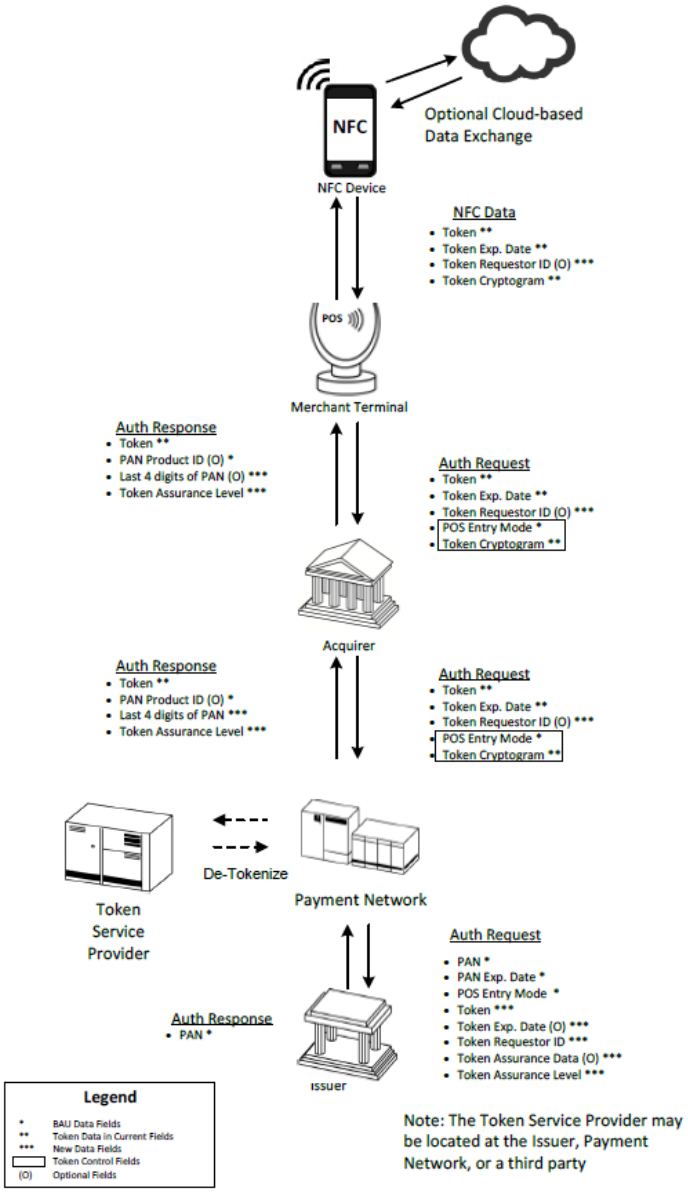


Figura 3.1: Figura extraída da norma EMV em EMVCO [7] demonstrando os elementos que correspondem a uma rede de captura baseada em geração de tokens.

A estrutura utilizada consistirá de Linux (Ubuntu 14.04) como o emissor do token e controlador do processo e de MS (Mobile Station) Android (4.4) como Cliente.

A seguir será proposto o detalhamento deste conceito bem como os códigos que deverão ser implementados para uma futura implantação do sistema. Como método para geração segura do token, que consiste em sequência pseudo-aleatória, a proposta se baseia na utilização do PRNG já mencionado como Fortuna. Uma proposta de código e testes de validação deste algoritmo também estão previstos ao longo deste trabalho.

A figura 3.2 apresenta a Topologia proposta acima.

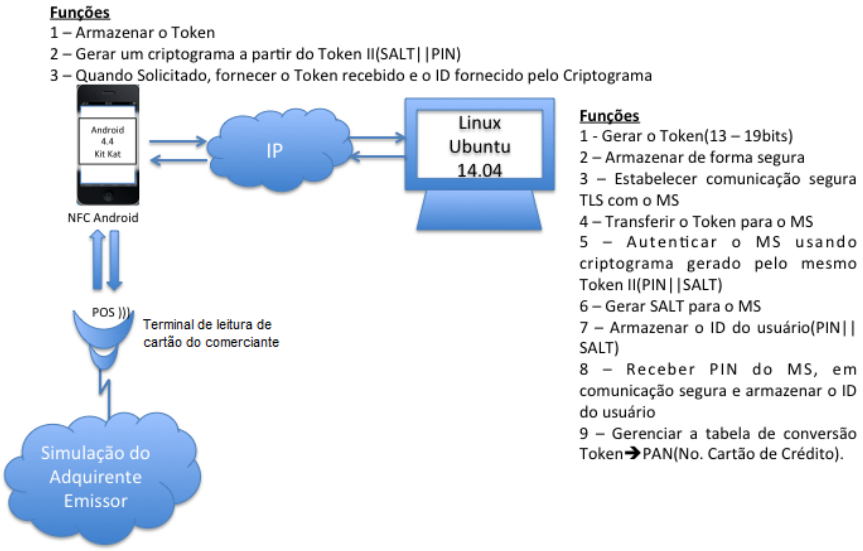


Figura 3.2: Topologia da proposta apresentada para implementação da Gestão da Identidade do usuário baseada em geração de tokens.

Para utilização das APIs (*Application Programming Interface*) no Android, foi escolhido o projeto desenvolvido pelo Google para utilização de linguagens baseadas em *Script* (tais como Python, Pearl, *script shell*, etc.) possam ser executadas em ambiente de aplicações Android ANDROID SCRIPTING [56]. Mais informações sobre este pacote de solução podem ser obtidas em C.1.

3.1.1 Benefícios da Geração de Tokens

Os maiores benefícios da abordagem de tokens sobre a propagação e/ou armazenamento da Identidade do usuário durante o processo de transação são:

1. Redução das possibilidades de Fraudes para as emissoras de cartão e para os portadores de cartões.

2. Os adquirentes e comerciantes serão beneficiados pela menor ocorrência de ataques online e vazamentos de informações críticas dos usuários, uma vez que as informações baseadas em token estarão armazenadas fora de seus sistemas.

3. Os gestores das redes de captura das transações poderão adotar uma topologia com especificações mais evoluídas de segurança digital, o que permitirá reduzir significativamente os custos de manutenção da segurança.

Esta melhoria na segurança do processo é a principal motivação para a adoção da geração de tokens. Nos processos atuais (pagamentos com cartão Chip and PIN e compra *online* http), a parte frágil do processo é constantemente atacada para fins de roubo de identidade do usuário e ações de fraudes. As figuras 3.3 e 3.4 ilustram a dinâmica destes possíveis ataques.

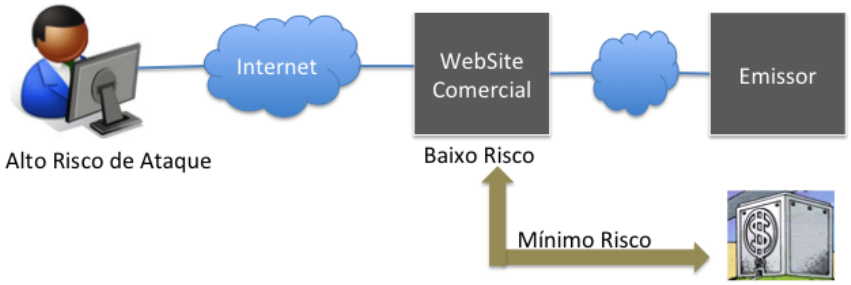


Figura 3.3: Demonstração da fragilidade dos processos atuais de pagamento: a interceptação e clonagem nas máquinas de pagamento.

Figura 3.4: Demonstração da fragilidade dos processos atuais de pagamento: ataque ao usuário residencial em modalidade *e-commerce*.

3.2 Implementação do Fortuna

Como afirmado anteriormente, o Fortuna será implementado no servidor principal (Linux Ubuntu). Nesta proposta, ele será o modelo de geração do token para identificação das transações financeiras que serão formuladas a partir da identidade individual do usuário da aplicação de pagamento. A seguir, o detalhamento de cada parte componente e sua implementação.

3.2.1 Acumulador

Uma das inovações mais importantes é o armazenador de entropia do Fortuna. Ele foi a grande evolução do *yarrow* FERGUSON e SCHNEIER [20] abandonando o estimador de entropia. Sua construção é relativamente simples:

- 32 filas de entropia, de $p_0, p_1 ... p_{31}$
- A taxa com que a entropia flui das suas fontes até o acumulador é denominada ρ, e sua unidade é bit/s.

Uma fila P_i coleta durante um dado tempo t, uma quantidade de bits equivalente a

$$2^t \rho t/32 \tag{3.1}$$

O tempo de recuperação uma dada p_i em caso de perda de entropia, por ataque ou não, é dado por $128 \leqslant 2^t \rho t/32 < 256$. Desta forma, chega-se ao tempo entre recargas de *pool* $2^t \rho t$ para recarregar $8192/\rho$ ou

$$2^t < \frac{8192}{\rho} \tag{3.2}$$

Este trabalho considera extremamente importante implementar um acumulador de entropia próprio para aplicação por duas razões:

1. O fato já mencionado que estudos comprovam que a entropia do sistema Linux nem sempre está disponível para que aplicações trabalhem em regime de segurança, especialmente após o *boot* do sistema. Isso se deve, basicamente, aos mecanismos de armazenamento da entropia que não atuam eficientemente para recuperação após "restart". A figura 3.5 apresenta o resultado das medições feitas por DING *et al.* [8] demonstrando a baixa disponibilidade de entropia imediatamente ($\approx 20s$) após a inicialização do sistema.

2. Em regime normal de trabalho, o servidor pode ser alvo de algum ataque que tenha como consequência a redução da entropia global disponível. Neste caso, a presença de acumuladores de entropia próprios podem garantir o correto funcionamento da aplicação até que uma ação seja tomada para conter o ataque. Tal ataque foi demonstrado em KAPLAN *et al.* [18].

3.2.2 Gerador

O gerador do Fortuna tem em seu núcleo uma implementação AES256 bits que, para este projeto, será o mesmo definidos na classe Python Crypto.Cipher com o módulo AES:

```
from Crypto.Cipher import AES
```

disponível na biblioteca Python. Esta classe é definida utilizando *CounTer Mode* (CTR) NIST [57].

Seus componentes são:

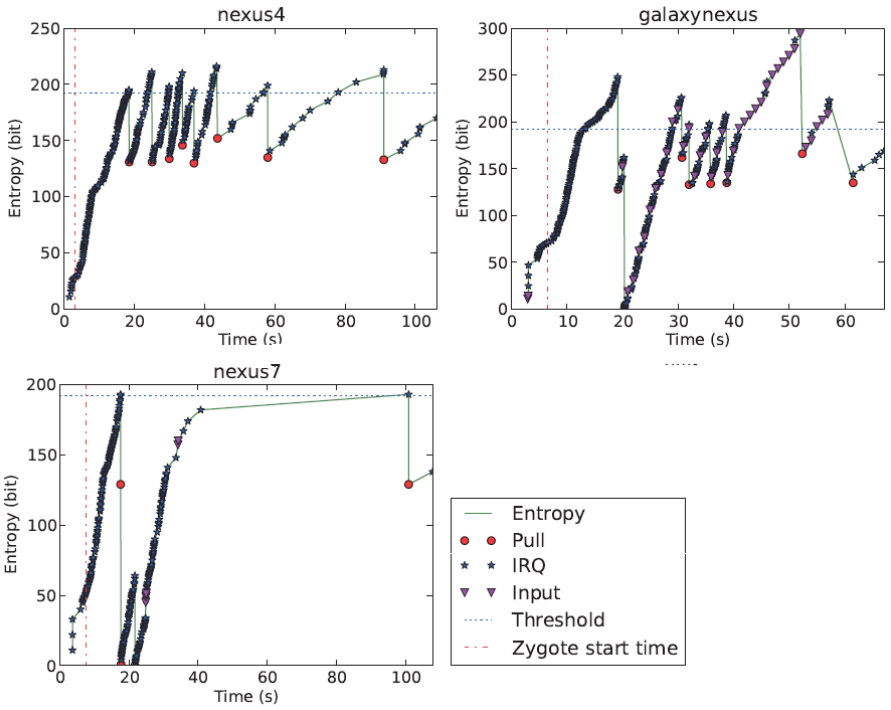


Figura 3.5: Gráficos apresentados em DING *et al.* [8] demonstrando a fragilidade do sistema em se recuperar após reinicialização

- Contador (C)128 bits.
- Chave criptográfica K de 256 bits.
- Variável de entrada (interno) k especificando o número de blocos.
- Variável de entrada n para especificar quantos bytes para cada solicitação.
- Saída de 128 bits, limitada a $2^{16} blocos$ de 128 bits ou $2^{20} bytes$. Esta limitação é uma forma de diminuir a probabilidade matemática de ataques e consequentes colisões FERGUSON e SCHNEIER [20].

Na figura 3.6 encontra-se uma representação gráfica do gerador do Fortuna.

3.2.3 Codificação

Para sua implementação, serão codificadas suas 04 etapas principais: Inicialização, *Reseed*, Geração de Blocos e Geração de Dados Aleatórios. Serão utilizadas classes Python FOUNDATION [58] adaptadas para simulação do gerador e acumulador do modelo Fortuna, adaptando o acumulador para receber o fluxo ρ de entropia nas 32 filas, a partir de */dev/random/* do Ubuntu.

A partir destas classes, retirar uma amostra de números aleatórios gerando um arquivo binário de 1Mbytes de tamanho, correspondente ao máximo gerado em

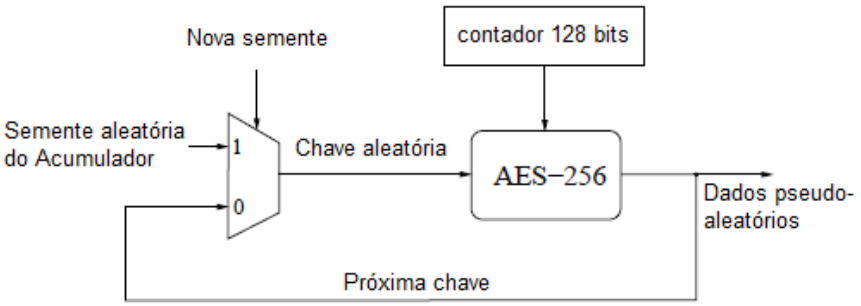


Figura 3.6: Ilustração do gerador Fortuna retirado de MCEVOY *et al.* [9]

uma solicitação (uma vez que o gerador está limitado a $2^{20} bytes$ ou $2^{16} blocos$ de saída).

Uma vez retirada as amostras, submeter ao teste de aleatoriedade no pacote do NIST, verificando os valores de *P-Values* e as faixas alcançadas. A codificação para a geração do pacote ficou conforme a seguir.

Inicialização: zerando a chave K e o contador C
saída: *G saída do gerador*

$$(k, C) \leftarrow (0, 0)$$
$$G \leftarrow (K, C)$$
$$\textbf{return } G$$

Ressed: realimentando o gerador com nova semente
Entradas: G estado do gerador e semente s

$$K \leftarrow SHA_d256(K \parallel s)$$
$$C = C + 1$$

Geração de Blocos
Entrada: G estado do gerador e k correspondente a quantidade de blocos a gerar

$$C \neq 0$$
$$r \leftarrow \epsilon \quad \textit{zerando a saida}$$
$$\textbf{for } i = 1 \textbf{ to } k \textbf{ do}$$
$$r \leftarrow r \parallel E(K, C)$$
$$C \leftarrow C + 1$$
$$\textbf{od}$$
$$\textbf{return } r$$

Geração de dados
Entrada: G estado do gerador e n número de bytes para gerar.

$$\textit{garantir } 0 \leq n \leq 2^{20}$$
$$r \leftarrow \textit{primeiros } n \textit{ bytes } (GeradordeBlocos(G, [n/16]))$$
$$\textit{substituir a chave}$$
$$K \leftarrow GeradordeBlocos(G, [n/16]))$$
$$\textbf{return } r$$

3.3 Geração da Identidade da Transação

3.3.1 SALT

A técnica de armazenamento seguro de senha mais implementada é a de *SALTED HASHING*, concebida originalmente por KELSEY *et al.* [21]. A técnica consiste em criar uma maior robustez no armazenamento de senhas. Neste trabalho, será aplicada esta técnica para evitar a necessidade de propagação da senha em formato *clear text* entre o MS e o servidor de tokens. O SALT é uma sequência pseudo-aleatória que, juntamente com o PIN de identificação do usuário, passará por um algoritmo de *hash*. Ao final deste processo, SALT e resultado final de SALT||PIN serão armazenados no servidor. Para manter a simetria da comunicação usuário-servidor, o SALT também é enviado para o MS a partir da comunicação segura utilizando TLS. Desta forma, ao receber o token, o usuário será solicitado a inserir o PIN de identificação.

Em seguida, é enviado para o servidor o resultado da operação abaixo assegurará que a identidade do usuário será atestada, validando um criptograma único para aquela transação associado aquele token emitido.

$$token \| (PIN \| SALT) \tag{3.3}$$

3.3.2 SALTED HASH

.

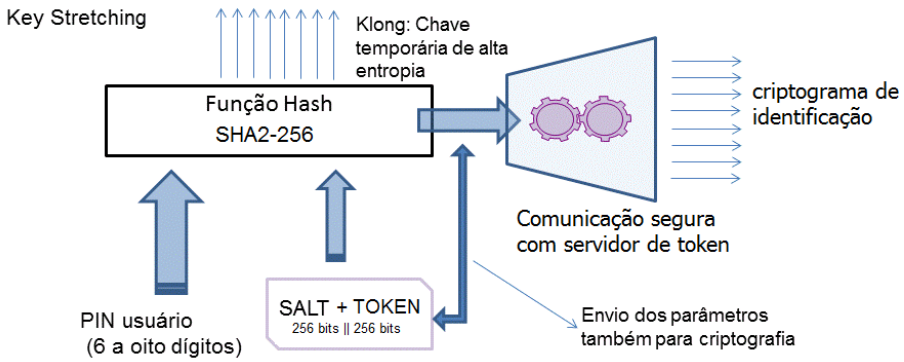


Figura 3.7: Método para geração de Identidade utilizando o token e Técnica de SALT

Uma vez gerado o ID do usuário, o mesmo é enviado para validação do servidor de token de acordo com a figura 3.8.

As etapas na execução da gestão de identidade remota baseada em token:

1. Estabelecer comunicação segura entre MS e Servidor utilizando bibliotecas SL4/TLS.

2. Servidor gera e envia token + SALT. Segundo norma EMV uma sequência de até 19 dígitos ASCII. Para os testes, foram considerados tokens de até 100Mbits, visando teste de performance.

3. MS utiliza as bibliotecas Android para armazenamento criptografado do token e SALT. A frequência de atualização do token no aparelho pode ser estabelecida para garantir constante atualização do processo.

4. MS gera o criptograma a partir das informações concatenadas (utilizando SHA256 como hash) entre PIN+SALT e token.

5. MS utiliza o criptograma para validar o usuário que está solicitando uma transação.

As etapas estão ilustradas na figura 3.9.

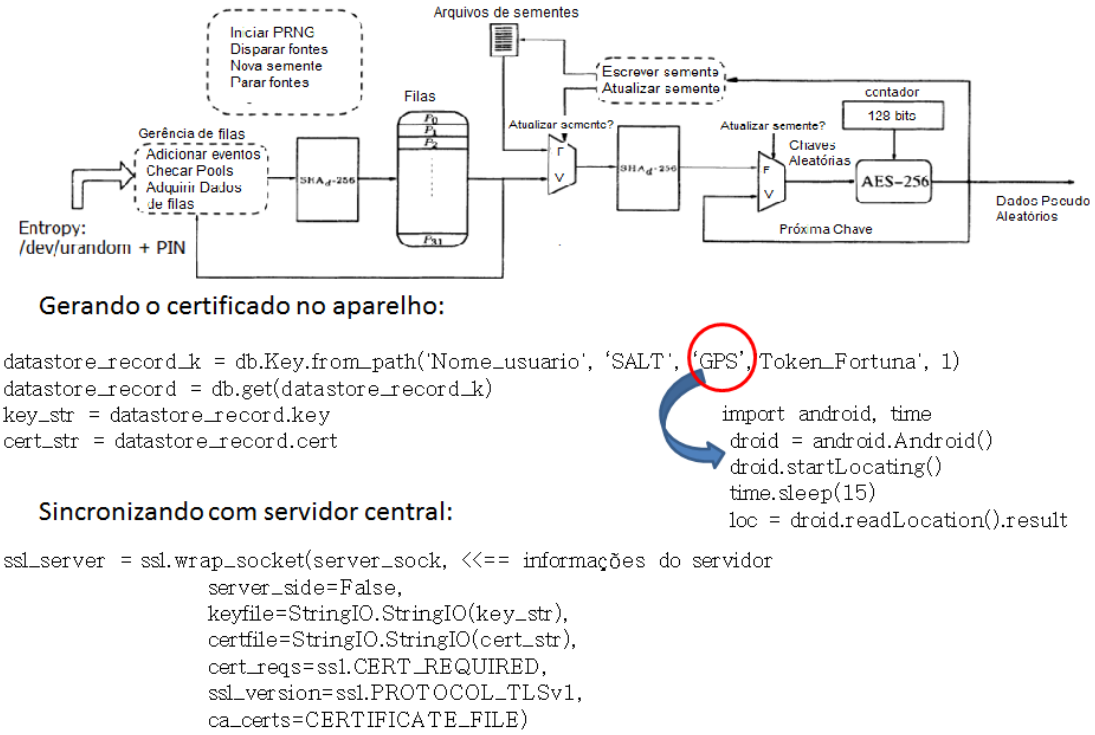


Figura 3.8: Detalhamento da geração do ID do usuário e posterior validação pelo servidor

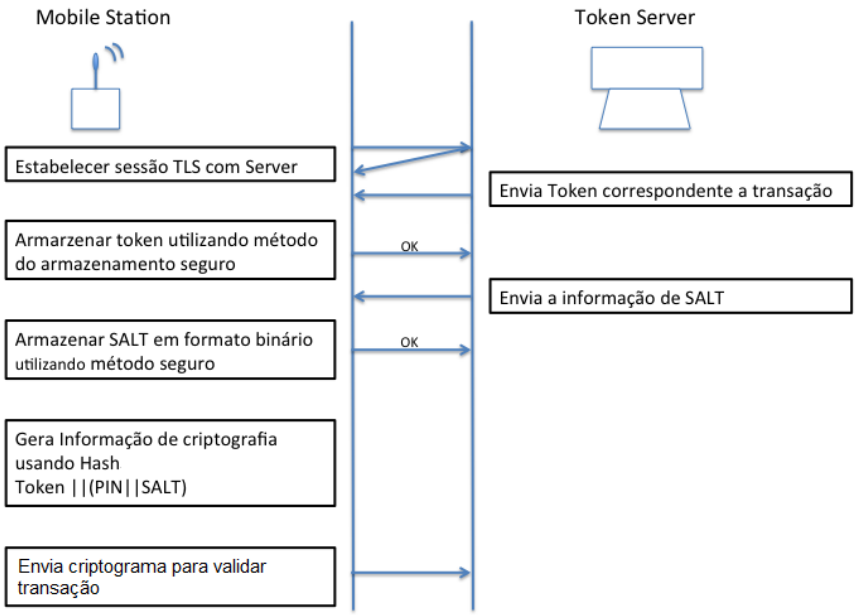


Figura 3.9: Arquitetura final destacando o fluxo de informações

Capítulo 4

Resultados e Discussões

4.1 Descrição dos testes

Para a validação da capacidade do gerador escolhido (AES em modo "*block count*") se manter em regime adequado (ou seja, gerando aleatoriedade com a *entropia* necessária), os testes mencionados na análise bibliográfica, os testes especificados pelo NIST se mostram mais adequados porque são tratados como padrão de mercado para implementações de segurança. O NIST recomenda a execução de 15 testes estatísticos, sendo alguns mais adequados para efeitos ilustrativos. Maior detalhamento destes testes ficará para o Apêndice A. Abaixo, segue a especificação das hipóteses adotadas conforme recomendação do instituto norte americano.

Tabela 4.1: Formulação das hipóteses de testes

Situação Verdadeira	**Conclusão**	
	Aceitar H_0	**Aceitar H_a (rejeitar H_0)**
Hipótese *Null*: H_0 é verdadeira e os dados são aleatórios.	Sem erro.	Erro Tipo I.
Hipótese Alternativa: H_a é verdadeira e os dados não são aleatórios.	Erro Tipo II.	Sem erro.

Complementando as especificações de testes, seguem os valores e limites especificados para intervalo de confiança e a precisão dos testes.

1. α: especifica o o *nível de significância* dos testes. Valor sugerido pelo *NIST* para testes estatísticos com geradores criptograficamente seguros é α= 0.01.

2. Massa gerada para testes: serão amostrados pacotes binários do gerador com tamanhos variando entre 15Mbits, 50Mbits e 100Mbits subdivididos em 100

sequências binárias de testes, para cada pacote (o que resultará em n=100 amostras para a análise estatísticas dos algoritmos). Este tamanho de amostra é sugestivo para submeter o gerador ao estresse de testes, mas é muito superior ao regime nominal de carga.

Dos 15 testes especificados, serão validados graficamente apenas 03 considerados mais importantes segundo L'ECUYER e SIMARD [23]. Teste da cadeia mais longa de "1's"em um bloco , Espectro Discreto da Transformada de Fourier e Teste de Entropia Aproximada. Todos, com maior detalhamento no Anexo A.

Execução dos testes:

```
host$ time dd if=gerador_aes of=nist_fortuna_15.dat bs=15M count=1

15728640 bytes (16 MB) copiados, 1,65142 s, 9,5 MB/s
real 0m1.653s
user 0m0.000s
sys 0m1.648s

$ time dd if=gerador_aes of=nist_fortuna_50.dat bs=15M count=1

15728640 bytes (16 MB) copiados, 1,71128 s, 9,2 MB/s
real 0m1.713s
user 0m0.000s
sys 0m1.708s

host$ time dd if=gerador_aes of=nist_fortuna_100.dat bs=100M count=1

104857600 bytes (105 MB) copiados, 11,604 s, 9,0 MB/s
real 0m11.607s
user 0m0.000s
sys 0m11.076s
```

O algoritmo utilizado para geração dos testes computa a partir das amostras os valores de Desvio padrão e média amostral. A partir destes valores, são obtidos os valores de referência Z da normal padrão $N(0,1)$ com os quais a confirmação da hipótese H_0 é aceita ou rejeitada.

As figuras a seguir demonstram a saída dos testes do NIST especificados para aferição do gerador de aleatoriedade obtido a partir da implementação descrita anteriormente.

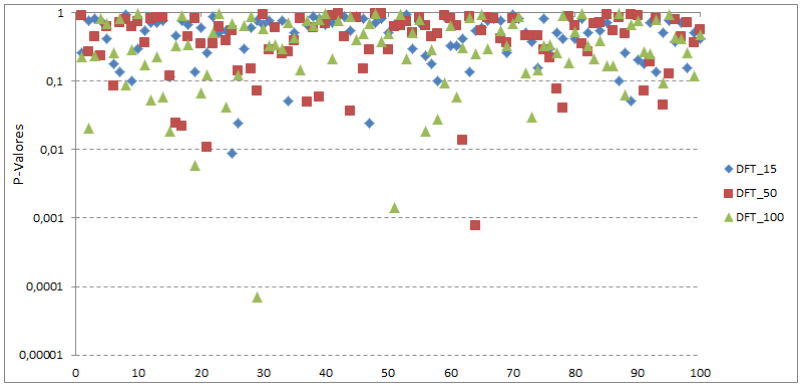


Figura 4.1: Resultado dos testes de Espectro de Fourier

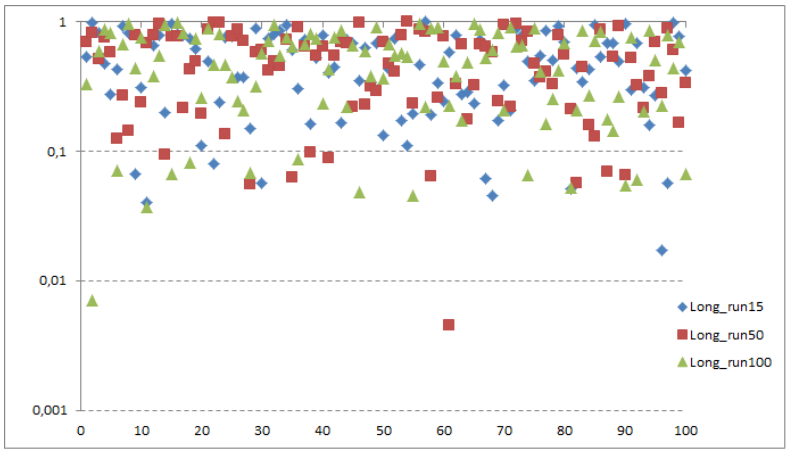


Figura 4.2: Resultado dos testes de Maior cadeia de 1

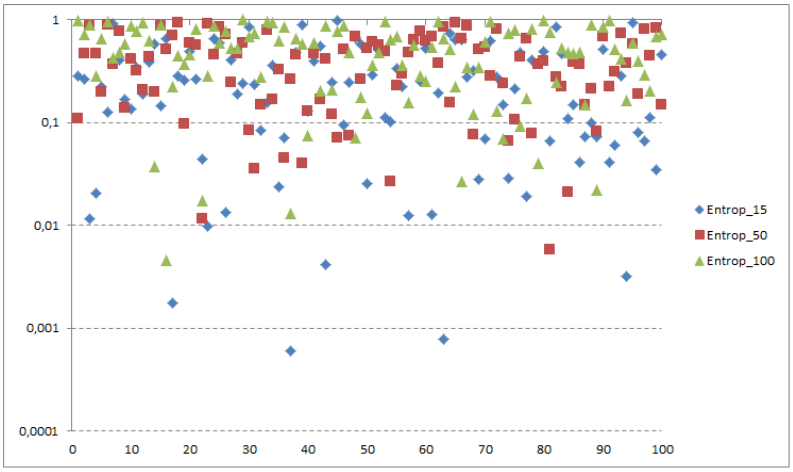


Figura 4.3: Resultado dos testes de Entropia Aproximada

Como já era esperado, a distribuição dos P-Valores dentro do intervalo de confiança de $1-\alpha$ se deu de forma inquestionável. Mais de 99% dos P-Valores estão acima de $\alpha = 0.01$ o que, estatisticamente, valida a hipótese H_0 de aleatoriedade das amostras.

Um resultado obviamente esperado uma vez que se utilizou algoritmos reconhecidamente eficientes como AES e SHA256. Mas havia a necessidade de demonstrar a eficiência da arquitetura proposta desenvolvida com a utilização de bibliotecas Python OpenSource.

Tabela 4.2: Resultado final NIST amostra 15 Mbits

RESULTS FOR THE UNIFORMITY OF P-VALUES AND THE PROPORTION OF PASSING SEQUENCES

generator is <data/nist_fortuna_15.dat>

C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	P-VALUE	PROPORTION	STATISTICAL TEST
11	14	11	12	6	11	10	9	10	6	0.779188	99/100	Frequency
4	7	14	6	14	7	14	12	12	10	0.181557	100/100	BlockFrequency
9	12	13	14	9	8	10	10	6	9	0.816537	99/100	CumulativeSums
13	12	5	14	6	12	12	9	10	7	0.455937	98/100	CumulativeSums
15	7	9	14	10	9	9	8	13	6	0.514124	97/100	Runs
9	12	9	11	12	6	10	14	8	9	0.851383	100/100	LongestRun
9	6	14	8	13	4	11	14	11	10	0.350485	100/100	Rank
8	10	10	4	6	16	6	19	15	6	0.006196	99/100	FFT

Tabela 4.3: Resultado final NIST amostra 50 Mbits

RESULTS FOR THE UNIFORMITY OF P-VALUES AND THE PROPORTION OF PASSING SEQUENCES

generator is <data/nist_fortuna_50.dat>

C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	P-VALUE	PROPORTION	STATISTICAL TEST
7	9	11	9	10	16	8	13	8	9	0.678686	99/100	Frequency
8	5	12	8	12	6	13	11	9	16	0.319084	99/100	BlockFrequency
9	8	7	10	15	11	14	8	6	12	0.534146	99/100	CumulativeSums
9	10	14	10	13	7	9	7	8	13	0.759756	99/100	CumulativeSums
12	8	11	15	10	10	7	9	14	4	0.383827	100/100	Runs
15	9	6	11	11	12	8	7	13	8	0.595549	100/100	LongestRun
14	7	15	6	11	7	7	10	9	14	0.334538	98/100	Rank
11	12	11	7	8	12	7	6	15	11	0.595549	99/100	FFT

Estas tabelas são a saída padrão dos testes disponibilizados pelo NIST para avaliação de algoritmos de aleatoriedade. Como foi usada 100 amostras de cada arquivo (15, 50 e 100 Mbits de tamanho), o número de avaliações positivas de cada amostra deve ser 96 em 100 amostras. Todos os testes foram positivos neste critério.

Importante uma ressalva: para o Android, não é tão crítica a condição apontada pelos estudos de DODIS *et al.* [17], KAPLAN *et al.* [18] e DING *et al.* [8]. Uma vez que o grande problema apontado por estes estudos foi a baixa entropia ime-

Tabela 4.4: Resultado final NIST amostra 100 Mbits

RESULTS FOR THE UNIFORMITY OF P-VALUES AND THE PROPORTION OF PASSING SEQUENCES

generator is <data/nist_fortuna_100.dat>

C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	P-VALUE	PROPORTION	STATISTICAL TEST
10	6	15	11	8	14	6	8	13	9	0.419021	98/100	Frequency
11	15	11	6	11	7	11	10	9	9	0.779188	98/100	BlockFrequency
12	8	12	10	9	11	15	5	10	8	0.657933	97/100	CumulativeSums
12	1	11	14	10	11	8	9	18	6	0.026948	98/100	CumulativeSums
5	11	8	14	10	14	13	7	7	11	0.437274	100/100	Runs
14	4	14	7	9	10	9	10	14	9	0.383827	99/100	LongestRun
8	16	7	10	8	7	15	14	6	9	0.213309	100/100	Rank
17	10	16	11	7	5	8	7	6	13	0.071177	97/100	FFT

diatamente após a inicialização do sistema, é improvável que algum serviço crítico de segurança seja utilizado até 20 ms após o *boot*. Durante este período, o usuário ainda está iniciando sua interação com o sistema móvel e não teria tempo hábil para iniciar nenhuma aplicação de compra, por exemplo.

4.2 Comunicação Segura TLS

Utilizando as bibliotecas SL4A disponíveis para o aparelho Android utilizado, foi possível utilizar as APIs adequadas para o estabelecimento de comunicação segura com o servidor de token. As API's foram suficientes para a construção de um protótipo. Mas, não são recomendadas para uma versão comercial ou testes de carga. Para tanto, a recomendação é a construção de código utilizando as API's nativas Java do sistema.

O teste de carga gerando as interações mencionadas no capítulo 3 na seção de SALT, não afetaram significativamente a performance do aparelho Android utilizado (Samsung Galaxy IV com versão 4.4 KitKat).

4.3 Métricas para avaliações adicionais

Como relatado, foram avaliadas possíveis impactos tanto no aparelho móvel quanto no servidor gerador de token, não sendo percebido nenhum impacto significativo dada a natureza dos testes (que não implementam todas as funcionalidades necessárias para um sistema comercial) e o baixo volume de solicitações.

Além disso, para uma real comparação desta proposta com as soluções de pagamentos em uso no mercado, seria necessária a utilização de uma infra-estrutura mais ampla, com a presença de recursos como equipamentos POS e TAD (leitores

de cartão) e as mesmas características de comunicação como linhas de baixa velocidade (1200Kbps, GPRS) normalmente utilizadas por estes equipamentos.

Como sugestões, os testes abaixo poderiam direcionar de forma mais eficiente uma comparação entre tecnologias, permitindo uma melhor consideração de viabilidade de uma sobre a outra.

- Consumo de tráfego por transação (Bytes/transação): é fato que o atual sistema de pagamento eletrônico consome pouca banda de dados, tendo os provedores até a possibilidade de utilizarem redes de tecnologia legada, como *frame-relay* e outras redes determinísticas baseadas em sub-taxas tais como 1200Kps, 9600Kbps. A migração destes serviços para aparelhos móveis não demandará necessariamente mais banda do que qualquer aplicativo amplamente utilizado para redes sociais,correio eletrônico, etc. Mas pode significar impactos em volume de processamento de bilhetagem e maior capacidade de redes em provedores de serviço.

- Latência na execução de transações (segundos/transação): esta métrica é importante porque, em geral, os usuários são bastante sensíveis a demora para realização de tarefas simples como pagamentos. Uma piora neste requisito pode significar a perda da janela de oportunidade para uma migração tecnológica como proposta neste trabalho.

- Custo por transação (R$ /transação): uma vez dimensionada a infra necessária para tendo-se em vista o volume de usuários que serão atendidos e o consequente volume de transações, esta estimativa é necessária como critério de decisão visando implementar uma migração tecnológica para os administradores do serviço.

Outros testes comparativos que podem ser implementados aplicando-se boas práticas de mercado voltado para a segurança da informação mais especificamente.

- Teste de interceptação de informações (*eavesdropping*): como mencionado em 2.2.13, algumas informações críticas podem ser roubadas do atual sistema com algum esforço e conhecimento da tecnologia. Um teste comparativo seria interceptar o fluxo de transação a partir da aplicação móvel (utilizando um monitorador de trafego como um *sniffer* (*wireshark*[1], por exemplo) registrando se alguma informação relevante (a partir do cabeçalho, como exemplo de informação aberta) poderia permitir a reconstrução de alguma credencial crítica do usuário ou aplicações envolvidas.

[1] www.wireshark.org

- Avaliação de boas práticas de segurança: existe um conjunto de orientações em relação a segurança da informação praticadas e auditadas no mercado. O melhor exemplo é a "Procedimento de Auditoria de Segurança - Indústria de Cartões de Pagamentos PCI"mantida pela *Security Standard Council*[2]. Nesta norma de auditoria adotada como *padrão de facto* pels prestadoras de serviços financeiros, encontram-se metodologias de avaliação e documentação de segurança, desde infra-estrutura, passando por pessoal e até aplicações. A emissão de um relatório de aderência da proposta feita nesta dissertação poderia elucidar uma comparação real com o sistema de mercado em vigor, bem como orientar para novas melhorias e evoluções futuras.

- Testes que ataquem a entropia dos equipamentos empregados como POS e TAD (leitores de cartão): analisando se seriam possíveis ataques como os mencionados em 2.4.3 e se existem implementações de contorno nos equipamentos em uso hoje nos principais pontos comerciais.

[2]www.pt.pcisecuritystandards.org

Capítulo 5

Conclusões

A segurança nas transações financeiras é de fato crítica para o correto funcionamento de nossa sociedade. É esperado que trabalhos como esta dissertação ocorram com maior frequência no futuro, demandadas principalmente pela necessidade de virtualização de serviços e funções hoje obtidas por meios físicos (hardware).

Esta dissertação procurou também exercer um papel de alerta para nossa sociedade tendo em vista que existe muita mistificação do assunto segurança da informação. De fato, dada a complexidade, este tema não é de fácil absorção por parte da nossa sociedade mas ao mesmo tempo ele não pode ser negligenciado. Outro fato relevante, é que a segurança da informação é mais frágil no lado do processo em que existe um ser humano executando alguma etapa crítica (digitando seu cartão de crédito, deslocando-se com suas credenciais, etc.). É esperado pois a natureza humana possui diversas facetas aleatórias e, desa aleatoriedade, podem surgir falhas.

O material apresentado nesta dissertação procura agregar segurança justamente na etapa crítica do processo de pagamento: a natureza humana envolvida neste ato. O método proposto para gestão da identidade diminui sensivelmente a probabilidade de se furtar informações relevantes que poderiam levar a fraudes.

5.1 Considerações Finais

As codificações propostas no trabalho demonstraram a eficiência do método mas precisam ser melhor exploradas quanto a performance e viabilidade comercial. Esta dissertação não teve como foco apresentar um resultado final que possa sustentar uma implementação comercial, por exemplo, mas apenas um modelo real e factível para chegar-se a um resultado aceitável. Em caso de interesses na implementação deste trabalho, deve-se ter em vista preocupações como linguagem, estrutura física, etc.

Os testes implementados foram suficientes para a validação tecnológica do *smartphone* Android executando transações financeiras de forma segura. As bibliotecas disponíveis estão atualizadas em relação as melhores práticas e existe grande agilidade dos gestores do sistema operacional para inserir atualizações e correções quando necessárias. A utilização do Linux Ubuntu não apresentou nenhuma característica que necessite ser considerada como impeditiva (fragilidade, ausência de pacotes, etc.) exceto a mencionada latência para a disponibilização de entropia para o sistema logo após o *restart* do sistema. A opção por uma construção utilizando pacotes *OpenSource* é apresentada aqui como factível, apesar da existência de certo preconceito com estes pacotes principalmente tratando-se de universo financeiro.

Para uma implantação sistêmica de solução, há necessidade de se avaliar toda a infra disponível e aplicação das métricas sugeridas como comparação com as opções de mercado existentes. Existem padrões de avaliação (auditoria) de segurança e outros itens críticos para o funcionamento da solução, de modo que neste aspecto não há nenhuma conveniência de inovação. Ao contrário, como regra geral deve-se seguir estes modelos *de facto* para se evitar qualquer rejeição quando de uma perspectiva comercial.

A proposta apresentada neste estudo pode facilmente ser adaptada para aplicações de cunho social, tais como distribuição de micro-créditos tais como Bolsa Família e semelhantes. É notável a existência de fraudes e desvios financeiros no sistema governamental, de modo que uma melhor automação e aplicação de disciplinas mais rígidas de segurança e gestão possam trazer ganhos significativos. Em termos de escala, para se implementar tal solução como alternativa viável em nível regional (atendendo uma população de municípios mais pobres, por exemplo) os investimentos necessários recairiam em maior peso sobre a compra dos aparelhos móveis individuais. A infra-estrutura de geração de tokens seria centralizada e, uma vez instalada em algum órgão gestor de tecnologia (SERPRO[1], por exemplo) cresceria de forma controlada com a demanda de novos usuários - que, em uma perspectiva correta e esperada de um programa de benefício social, tende a diminuir com o tempo. Outro dado relevante é que se pode adquirir uma plataforma móvel *smart* por preços abaixo dos aparelhos *top* de linha no mercado. O preço médio encontrado em aparelhos que possuam as funcionalidade mínimas necessárias (comunicação segura, NFC, etc.) partem de valores como $\mathrm{R\$}300,00$. A exemplo de países africanos que conseguiram grande penetração de programas

[1]Serviço Federal de Processamento de Dados

de microcrédito a partir de aplicações móveis, o Brasil teria grande retorno caso também se enveredasse por este caminho - tenho o necessário apoio governamental.

Por fim, algumas considerações sobre o uso do NFC como solução de aproximação. Com a entrada da Apple no NFC Forum e a implementação do seu serviço de pagamento baseado em iOS, espera-se grande uso e popularização de serviços com esta tecnologia disponível nos aparelhos móveis. Entretanto, dada a total possibilidade de se implementar soluções seguras capazes de utilizarem as redes 3G, LTE sem perdas de segurança (com o emprego de comunicações criptografadas, claro) este trabalho questiona a real necessidade de aproximação como forma de pagamento. Se ambos os agentes envolvidos em um processo de compra, o vendedor e comprador, estão munidos de plataformas móveis capazes de suportarem aplicações de pagamentos, por qual motivo não executar toda a transação apenas com trocas de informações e identidades *on line*, sem a necessidade de existência de um aparelho fisicamente no local (POS ou TAD) para executar a transação? Isso não apenas simplificaria o processo e diminuiria o preço pago pelo serviço (uma vez que a infra seria menor), mas também reduziria os riscos de fraudes pelos motivos apresentados no que diz respeito a segurança dos aparelhos envolvidos na leitura de cartões. Onde existe a oferta de comunicação sem fio utilizando a rede móvel das grandes operadoras, poder-se-ia implantar uma solução sem a necessidade de qualquer equipamento adicional.

5.2 Trabalhos Futuros

Conforme mencionado, verificar as melhores práticas para a implementação comercial deste modelo pode ser uma sequência viável para esta dissertação. Além disso, outras frentes tecnológicas estão abordando o mercado e requerendo maior flexibilidade na autenticação dos usuários.

Outro ponto relevante, é que o conceito de gestão de identidade não é aplicável apenas em métodos de pagamentos, podendo também ser utilizado em outras frentes:

- Internet das coisas (IoT). Com o crescimento de dispositivos móveis utilizando aplicações diversas, serão necessárias formas de controle de segurança para aplicações e usuários.
- Controle de acesso físico eletrônico. O emprego de soluções como crachá ou qualquer outro meio de credenciamento físico podem facilmente ser substituído. Com o benefício de migrarem para uma plataforma amplamente difundida e, normalmente, presente junto ao usuário em qualquer circunstância.

- GED: Gestão Eletrônica de Documentação. O controle de acesso a documentos críticos em organizações pode migrar para as plataformas móveis pessoais de posse dos usuários. Ao solicitarem acesso a qualquer documento de uso restrito, o usuário teria sua identidade questionada via aplicações embarcadas em seus aparelhos móveis, de modo que os níveis de autorização e acesso pudessem ser verificados pela plataforma gestora com bastante segurança e confiabilidade.

- Como já mencionado, a aplicação em serviços sociais para distribuição de micro-créditos, com a vantagem de se utilizar uma plataforma moderna e segura, que possa agregar aos processos de gestão informações adicionais dos usuários, tais como posição geográficas, informações em tempo real sobre a segurança (pode ser relevante em áreas de risco), acompanhamento do perfil consumidor de pessoas em situação de extrema pobreza.

Referências Bibliográficas

[1] IDC. *Smartphone OS Market Share, Q3 2014*. Relatório técnico, Acessado Julho, http://www.idc.com/prodserv/smartphone-os-market-share.jsp, 2014.

[2] NETO, F. M. *Mobile Payment e o caso da Paggo no Brasil*. M.Sc. dissertação, Escola de Administração de Empresas de São Paulo, São Paulo, SP, 2013.

[3] PAAR, C., PELZL, J. *Understanding Cryptography - A Textbook for Students and Practitioners*. Springer, 2010. ISBN: 978-3-642-04100-6. doi: http://dx.doi.org/10.1007/978-3-642-04101-3.

[4] PETITCOLAS, F., ANDERSON, R., KUHN, M. ""Information hiding - a survey"", *Proceedings of the IEEE*, v. 87, n. 7, pp. 1062 –1078, jul 1999. ISSN: 0018-9219. doi: 10.1109/5.771065.

[5] CAYIRCI, E., RONG, C. *Security in Wireless Ad Hoc and Sensor Networks*. Wiley, 2008. ISBN: 9780470516775. Disponível em: <`http://books.google.com.br/books?id=3EvhTrocBZUC`>.

[6] PUJOLLE, G., AISSAOUI, H., URIEN, P. *Global Identity Management of Virtual Machines Based on Remote Secure Elements*. In: Report hal-01018068, L'archive ouverte pluridisciplinaire HAL, Paris, 2014.

[7] EMVCO. *American Express, Discover, JCB, MasterCard, UnionPay, and Visa*. http://emvco.com/, Agosto, 2014.

[8] DING, Y., PENG, Z., ZHOU, Y., et al. "Android low entropy demystified". In: *Communications (ICC), 2014 IEEE International Conference on*, pp. 659–664, June 2014. doi: 10.1109/ICC.2014.6883394.

[9] MCEVOY, R., CURRAN, J., COTTER, P., et al. "Fortuna: Cryptographically Secure Pseudo-Random Number Generation In Software And Hardware". In: *Irish Signals and Systems Conference, 2006. IET*, pp. 457–462, June 2006.

[10] LATOUR, B., BENEDETTI, I., DE PAULA ASSIS, J. *Ciência em ação*. Biblioteca básica. São Paulo, SP, Brasil, Editora UNESP, 2000. ISBN: 9788571392656.

[11] ANDROID. *Google Inc.* https://www.android.com/, Junho, 2014.

[12] BITCOIN. *Bitcoin Foundation*. https://bitcoin.org/en/, September, 2014.

[13] ZDNET. *How hackers stole millions of credit card records from Target*. Relatório técnico, Zdnet.com, http://www.zdnet.com/article/how-hackers-stole-millions-of-credit-card-records-from-target/, 2014.

[14] G1, N. *Esquema de clonagem de cartões é desarticulado em Porto Alegre*. Relatório técnico, G1.com, http://g1.globo.com/rs/rio-grande-do-sul/noticia/2014/09/esquema-de-clonagem-de-cartoes-e-desarticulado-em-porto-alegre.html, 2014.

[15] KREUTZ, D., FEITOSA, E. "Identity Providers-as-a-Service built as Cloud-of-Clouds: challenges and opportunities". In: *Federated Conference on Computer Science and Information Systems*, p. 101–108, Poland, set. 2014.

[16] GUTTERMAN, Z., PINKAS, B., REINMAN, T. "Analysis of the Linux Random Number Generator". In: *Proceedings of the 2006 IEEE Symposium on Security and Privacy*, SP '06, pp. 371–385, Washington, DC, USA, 2006. IEEE Computer Society. ISBN: 0-7695-2574-1. doi: 10.1109/SP.2006.5. Disponível em: <http://dx.doi.org/10.1109/SP.2006.5>.

[17] DODIS, Y., POINTCHEVAL, D., RUHAULT, S., et al. "Security Analysis of Pseudo-random Number Generators with Input: /Dev/Random is Not Robust". In: *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, CCS '13, pp. 647–658, New York, NY, USA, 2013. ACM. ISBN: 978-1-4503-2477-9. doi: 10.1145/2508859.2516653. Disponível em: <http://doi.acm.org/10.1145/2508859.2516653>.

[18] KAPLAN, D., KEDMI, S., HAY, R., et al. "Attacking the Linux PRNG On Android: Weaknesses in Seeding of Entropic Pools and Low Boot-Time Entropy". In: *8th USENIX Workshop on Offensive Technologies (WOOT 14)*, San Diego, CA, ago. 2014. USENIX Association. Disponível em: <https://www.usenix.org/conference/woot14/workshop-program/presentation/kaplan>.

[19] SCHNEIER, B. *Applied cryptography: protocols, algorithms, and source code in C*. 2nd ed. New York, Wiley, 1996. ISBN: 0–471–12845–7.

[20] FERGUSON, N., SCHNEIER, B. *Practical Cryptography*. 1 ed. New York, NY, USA, John Wiley & Sons, Inc., 2003. ISBN: 0471223573.

[21] KELSEY, J., SCHNEIER, B., HALL, C., et al. "Secure Applications of Low-Entropy Keys". In: *Lecture Notes in Computer Science*, pp. 121–134. Springer-Verlag, 1998.

[22] L'ECUYER, P. "Random Number Generation". In: Gentle, J. E., Härdle, W., Mori, Y. (Eds.), *Handbook of Computational Statistics*, 1 ed., cap. 2, Berlim, German, Springer, 2004.

[23] L'ECUYER, P., SIMARD, R. "TestU01: A C library for empirical testing of random number generators", *ACM Transactions on Mathematical Software (TOMS)*, v. 33, n. 4, pp. 22, 2007.

[24] SIBLEY, E. H., PARK, S. K., KEITH, et al. "RANDOM NUMBER GEUERATORS: GOOD ONES ARE HARD TO FIND", *Communications of the ACM*, v. 31, n. 10, pp. 1192–1201, out. 1988.

[25] MARKOWSKY, G. "The Sad History of Random Bits", *Journal of Cyber Security*, v. 3, n. 1, pp. 1–26, fev. 2014.

[26] BOND, M., CHOUDARY, O., MURDOCH, S. J., et al. "Chip and Skim: cloning EMV cards with the pre-play attack", *IEEE Symposium on Security and Privacy - San Jose, CA*, v. 1, n. 1, pp. 18–21, maio 2014.

[27] ROLAND, M. "Contact versus Contactless Smartcards". In: Sebastian, M., Küper, A., Raake, A. (Eds.), *Security Issues in Mobile NFC Devices*, 1 ed., cap. 2, Hagenberg, Austria, Springer, 2015.

[28] ELENKOV, N. *Android Security Internals: An In-Depth Guide to Android's Security Architecture*. 1 ed. San Francisco, CA, No Starch Press, 2014.

[29] LERNER, T. *Mobile Payment*. 1 ed. Germany, Springer Fachmedien Wiesbaden, 2013.

[30] DINIZ, E. H., ALBUQUERQUE, J. P., CERNEV, A. K. "Mobile Money and Payment: a literature review based on academic and practitioner-oriented publications (2001-2011)". In: *Proceedings of SIG GlobDev Fourth Annual Workshop*, pp. 1–27, Shanghai, China, Dezember 2011.

[31] 3GPP. *Technical realization of the Short Message Service (SMS)*. 3GPP TS 23.040 Release 6, SMS/GSM/UMTS, http://www.3gpp.org, 2006-2010.

[32] MASSOTH, M., BINGEL, T. "Performance of Different Mobile Payment Service Concepts Compared with a NFC-Based Solution", *Internet and Web Applications and Services, International Conference on*, v. 0, pp. 205–210, 2009. doi: http://doi.ieeecomputersociety.org/10.1109/ICIW.2009.112.

[33] 3GPP2. *Unstructured Supplementary Service Data (USSD) Service Options for Spread Spectrum Systems: Service Options 78 and 79*. 3GPP2 C.S0105-0 v1.0 Release 1, 3GPP2-WG, www.3gpp2.org, 2012.

[34] GRECAS, C. F., MANIATIS, S. I., VENIERIS, I. S. "Introduction of the Asymmetric Cryptography in GSM, GPRS, UMTS, and Its Public Key Infrastructure Integration", *Mob. Netw. Appl.*, v. 8, n. 2, pp. 145–150, abr. 2003. ISSN: 1383-469X. doi: 10.1023/A:1022285130956. Disponível em: <http://dx.doi.org/10.1023/A:1022285130956>.

[35] 3GPP. *Technical Specification Group Services and System Aspects, 3G Security, Security Threats and Requirements*. 3GPP TS 21.133 3.2.0, SMS/GSM/UMTS, http://www.3gpp.org, 2001-2012.

[36] FORUM, N. *NFC Specifications*. Relatório técnico, Acessado Setembro, www.nfc-forum-org, 2014.

[37] ISO. *International Organization for Standardization*. Relatório técnico, Acessado Julho, http://www.iso.org/iso/home.html, 2014.

[38] POURALI, A., MALAKOOTI, M. V., YEKTAIE, M. H. "A Secure SMS Model in E-Commerce Payment using Combined AES and ECC Encryption Algorithms". In: *Proceedings of the International conference on Computing Technology and Information Management*, UAE, pp. 431–442, Dubai, 2014. SDIWC. ISBN: 978-0-9891305-5-4. Disponível em: <http://dx.doi.org/10.1109/SP.2006.5>.

[39] IGOE, T., COLEMAN, D., JEPSON, B. "Beginning NFC with Arduino, Android and PhoneGap". In: Eckert, E. R. G., Goldstein, R. J. (Eds.), *Measurements in Heat Transfer*, 2 ed., cap. 10, New York, USA, Hemisphere Publishing Corporation, 1976.

[40] LINCK, K., POUSTTCHI, K., WIEDEMANN, D. "SECURITY ISSUES IN MOBILE PAYMENT FROM THE CUSTOMER VIEWPOINT". In: *Proceedings of the 14th European Conference on Information Systems*, ECIS 2006, pp. 1–11, Göteborg, Schweden, 2006. University of Augsburg. Disponível em: <http://mpra.ub.uni-muenchen.de/2923/1/>.

[41] SCHNEIER, B. *Secrets and Lies: Digital Security in a Networked World*. John Wiley & Sons, 2004. ISBN: 0471453803.

[42] KATZ, J., LINDELL, Y. *Introduction to Modern Cryptography (Chapman & Hall/Crc Cryptography and Network Security Series)*. Maryland, USA, Chapman & Hall/CRC, 2007. ISBN: 1584885513.

[43] SHANNON, C. "Communication Theory of Secrecy Systems", *Bell System Technical Journal*, v. 28, n. 28, pp. 656–715, out. 1949. ISSN: 0001-02322. doi: 10.1145/359168.359176. Disponível em: <http://www.bellsystems.com>.

[44] RIVEST, R. L., SHAMIR, A., ADLEMAN, L. "A method for obtaining digital signatures and public-key cryptosystems", *Commun. ACM*, v. 21, n. 2, pp. 120–126, fev. 1978. ISSN: 0001-0782. doi: 10.1145/359340.359342. Disponível em: <http://doi.acm.org/10.1145/359340.359342>.

[45] EDNEY, J., ARBAUGH, W. *Real 802.11 security: Wi-Fi protected access and 802.11i*. Addison-Wesley, 2004. ISBN: 9780321136206. Disponível em: <http://books.google.com.br/books?id=aelSAAAAMAAJ>.

[46] GUPTA, V., KRISHNAMURTHY, S., FALOUTSOS, M. "Denial of service attacks at the MAC layer in wireless ad hoc networks". In: *MILCOM 2002. Proceedings*, v. 2, pp. 1118 – 1123 vol.2, oct. 2002. doi: 10.1109/MILCOM.2002.1179634.

[47] STUBBLEFIELD, A., IOANNIDIS, J., RUBIN, A. D. "Using the Fluhrer, Mantin, and Shamir Attack to Break WEP". pp. 17–22, 2001.

[48] FLUHRER, S. R., MANTIN, I., SHAMIR, A. "Weaknesses in the Key Scheduling Algorithm of RC4". In: *Revised Papers from the 8th Annual International Workshop on Selected Areas in Cryptography*, SAC '01, pp. 1–24, London, UK, UK, 2001. Springer-Verlag. ISBN: 3-540-43066-0. Disponível em: <http://dl.acm.org/citation.cfm?id=646557.694759>.

[49] YI, N. S., YI, S., KRAVETS, R. "MOCA : Mobile Certificate Authority for Wireless Ad Hoc". In: *2nd Annual PKI Research Workshop Program (PKI 03*, pp. 65–79.

[50] KENT, S., POLK, T. "Public Key Infrastructure Using X.509", *PKIX Working Group*, v. 1, pp. 1–55, 2005.

[51] DIFFIE, W., HELLMAN, M. "New directions in cryptography", *Information Theory, IEEE Transactions on*, v. 22, n. 6, pp. 644 – 654, nov 1976. ISSN: 0018-9448. doi: 10.1109/TIT.1976.1055638.

[52] SHAMIR, A. "How to share a secret", *Commun. ACM*, v. 22, n. 11, pp. 612–613, nov. 1979. ISSN: 0001-0782. doi: 10.1145/359168.359176. Disponível em: <http://doi.acm.org/10.1145/359168.359176>.

[53] SHANNON, C. E. "A Mathematical Theory of Communication", *SIGMOBILE Mob. Comput. Commun. Rev.*, v. 5, n. 1, pp. 3–55, jan. 2001. ISSN: 1559-1662. doi: 10.1145/584091.584093. Disponível em: <http://doi.acm.org/10.1145/584091.584093>.

[54] SCHILLER, J., CROCKER, S. "Randomness Requirements for Security". In: *BCP 106, RFC 4086*, 2005.

[55] NIST. *National Institute of Standards and Technology.* PDF RANDOM NUMBER GENERATION, NIST.gov, http://csrc.nist.gov/publications/nistpubs/800-22-rev1a/SP800-22rev1a.pdf, 2010.

[56] ANDROID SCRIPTING. *Scripting Layer for Android brings scripting languages to Android.* www Project Home, Google.com, https://code.google.com/p/android-scripting/, 2014.

[57] NIST. *National Institute of Standards and Technology.* PDF ADVANCED ENCRYPTION STANDARD (AES), NIST.gov, http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf, 2001.

[58] FOUNDATION, P. S. *Documentation.* Web, Python.org, https://www.python.org/doc, 2014.

Apêndice A

Especificação dos Testes

A.1 Avaliação dos testes empíricos

Para os testes executados com base nas orientações do NIST NIST [55] para avaliação de aleatoriedade em implementações de segurança que executam funções de geração aleatória de sequências binárias, algumas funções devem ser destacadas porque são recorrentes na utilização dos algoritmos dos testes.

A.1.1 Função Chi-Quadrado

$$\mathcal{X}^2 = \sum_i \frac{(o_i - e_i)}{e_i} \tag{A.1}$$

onde o_i e e_i são, respectivamente, os valores observados e esperados das frequências das medidas efetuadas nos testes.

A.1.2 Função Erro Complementar

$$erfc = \frac{2}{\sqrt{\pi}} \int_z^\infty e^x dx \tag{A.2}$$

Função Erro Complementar utilizado nos pacotes ANSI C[1], cuja função está contida no arquivo *math.h*. Esta biblioteca é incluída na instalação/compilação do pacote de testes.

A.1.3 Função Gamma

$$\Gamma(z) = \int_0^\infty x^{z-1} e^{-x} dx \qquad z > 0 \tag{A.3}$$

[1] *American National Standards Institute*

A.1.4 Função Gamma Incompleta

$$\mathcal{Q}(a,x) \equiv 1 - P(a,x) \equiv \frac{\Gamma(a,x)}{\Gamma(a)} \equiv \frac{1}{\Gamma(a)} \int_x^\infty e^{-t} t^{a-1} dt \tag{A.4}$$

Onde $\mathcal{Q}(a,0) = 1$ e $\mathcal{Q}(a,\infty) = 0$.
A função Gamma Incompleta é uma aproximação[2] utilizada na biblioteca C ANSI. Dependendo de seus valores a e x, a função gamma incompleta pode ser aproximada usando tanto frações contínuas ou séries.

A.2 Teste da cadeia mais longa de "1"em um bloco

O foco deste teste é a maior cadeia de "1's"dentro de blocos com tamanho M-bits. O propósito é determinar se o tamanho da maior cadeia de "1's"dentro das sequências binárias em teste é consistente com o tamanho da cadeia de "1's"esperada em uma sequência aleatória. Notar que uma irregularidade estatística nesta avaliação de tamanho de "1's"implica em igual irregularidade para a mais longa cadeia de "0's"também. Portanto, somente um teste é necessário.

O tamanho da maior sequência contígua de "1's"é uma característica para a determinação da aleatoriedade usada em testes. Os parâmetros do teste:

- n: tamanho da sequência binária em teste.
- N: quantidade de subdivisões que cada sequência original de tamanho n é particionada para testes.
- M: tamanho em bits de cada parte N, de modo que $n = N.M$.
- v_j: tamanho em bits da maior cadeia de "1's"de uma subdivisão de n, cujo intervalo é determinado pela quantidade de partes N. Ou seja, $v_1, ..., v_N$, sendo $j \, \epsilon \, [1, N]$.
- K: graus de liberdade da função χ^2. É determinado o valor de K em função de M e N. O NIST estabelece valores tabelados para limitar tempos de execução e processamento.

Tendo-se escolhido o tamanho n de cada sequência binária (o NIST sugere valores entre 128 bits e 750 Kbits), utiliza-se a tabela A.1 para a determinação dos

[2]M. Abramowitz e I. Stegun, Handbook of Mathematical Functions, Applied Mathematics Series. Vol. 55, Washington: National Bureau of Standards, 1964; reeditado em 1968 por Dover Publications, New York.

valores de k, M e N. Com estes valores.

Tabela A.1: Determinação dos valores M, N e K

Minimum *n*	***M***
128	8
6272	128
750,000	10^4

Determinação de M, a partir da escolha do tamanho máximo "n"em bits da sequência a ser testada

M	**K**	**N**
8	3	16
128	5	49
10^4	6	75

Determinação de K e N, a partir da inferência de M.

A probabilidade teórica de $\pi_0, \pi_1, ..., \pi_K$ é calculada pela equação A.5, tendo como entrada os valores de v, que correspondem ao número de "1's"contíguos em cada amostra de N. Esta probabilidade é obtida a partir das combinações possíveis do número de "1's"r obtidos em cada amosta (ao todo N amostras) e o tamanho máximo da cada amostra M. O valor de v_i é a quantidade esperada de "1's"de cada amostra, sendo este valor dependente das escolhas dos parâmetros acima. A quantidade esperada de "1's"é definida na tabela A.2: na tabela, é estabelecida a correspondência entre a quantidade de classes K e os valores esperados de "1's", assumindo a hipótese de aleatoriedade. O preenchimento destes valores é exemplificado abaixo.

$$\pi = P(v \leq r) = \sum_{r=0}^{K} \binom{M}{r} [v_r . \frac{1}{2^M}] \tag{A.5}$$

O Cálculo dos $P - Valores$ será feito a partir de

$$\frac{\int_{\mathcal{X}^2}^{\infty} x^{K/2-1} e^{-x/2} dx}{\Gamma(K/2) 2^{K/2}} = \mathcal{Q}(\frac{K}{2}, \frac{\mathcal{X}^2}{2}) \tag{A.6}$$

onde $\mathcal{X}^2$ é definido por

$$\mathcal{X}^2 = \sum_{i=0}^{K} \frac{(v_i - N\pi)^2}{N\pi_i} \tag{A.7}$$

Um exemplo prático do preenchimento de cada valor v_{K+1} segue abaixo. A partir dos valores de parâmetros escolhidos na tabela A.3, onde é demonstrada uma

Tabela A.2: Valores esperados para ocorrências de "1's"em cada v_{k+1}

$\mathbf{v_i}$	**M = 8**	**M = 128**	**M = 10^4**
$\mathbf{v_0}$	≤ 1	≤ 4	≤ 10
$\mathbf{v_1}$	2	5	11
$\mathbf{v_2}$	3	6	12
$\mathbf{v_3}$	≥ 4	7	13
$\mathbf{v_4}$		8	14
$\mathbf{v_5}$		≥ 9	15
$\mathbf{v_6}$			≥ 16

sequência de amostra de comprimento original $n = \epsilon = 128$. Observando para n os valores correspondentes de $M = 8, K = 3eN = 16$, é possível verificar que teremos grau de liberdade 3, mas com $v_0, ..., v_K$. Na coluna correspondente ao valor de $M = 8$ na tabela A.3, é possível verificar os valores esperados para cada v. Desta forma, procede-se a contagem em cada uma das subsequências $N = 16$, associando em cada v_K o número total de subsequências que atendem ao valor esperado tabelado. Com os valores de v_K calculam-se os valores de A.5 e A.7.

Tabela A.3: Determinação dos valores empíricos para v_{k+1}

Exemplo para $K = 3$ and $M = 8$:

(Entrada de bits) ε = 11001100000101010110110001001100111000000000001001
00110101010001000100111101011010000000110101111100
11001110011011011000010110010

(tamanho em bits) $n = 128$

(Processando)

Subblock	Max-Run	Subblock	Max-Run
11001100	(2)	00010101	(1)
01101100	(2)	01001100	(2)
11100000	(3)	00000010	(1)
01001101	(2)	01010001	(1)
00010011	(2)	11010110	(2)
10000000	(1)	11010111	(3)
11001100	(2)	11100110	(3)
11011000	(2)	10110010	(2)

Resultado: $v_0 = 4;\ v_1 = 9;\ v_2 = 3;\ v_4 = 0;$

A.3 Espectro Discreto da Transformada de Fourier

Este teste se baseia na transformada discreta de Fourier. O teste de Fourier detecta características periódicas em séries de bits que acusam qualquer desvio da aleatoriedade assumida como premissa.

Seja x_k o Késimo bit, onde k=1,...,n. Assuma que os bits são codificados como -1 ou $+1$.

$$S_j = \sum_{k=1}^{n} \exp(2\pi i(k-1)\frac{j}{n}) \tag{A.8}$$

Onde $\exp(2\pi ikj/n) = \cos(2\pi kj/n) + i\sin(2\pi kj/n)$, para j=0,..,1, $n-1$ e i=$\sqrt{-1}$. Devido a simetria entre os valores reais e os complexos na transformada, somente valores de 0 a $(n/2-1)$ são considerados.
Seja $|S_j|$ o módulo do número complexo S_j. Considerando a premissa de aleatoriedade da série x_i, um intervalo de confiança pode ser estabelecido nos valores de $|S_j|$. Mais especificamente, 95% de $\mod_j$ devem ser menores que $h = \sqrt{(\log\frac{1}{0.05})n}$. Um $P-Valor$ baseado nestas especificações surge de uma distribuição *binomial*.

Seja N_1 o número de picos, exceto pelo valor h. Somente $n/2$ picos são considerados. Seja $N_0 = .95N/2$ e $d = (N_1 - N_0)/\sqrt{n(0.95)(0.05)/4}$. O $P-Valor$ é

$$2(1-\phi(|d|)) = erfc(\frac{|d|}{\sqrt{2}}) \tag{A.9}$$

Onde $\phi(x)$ é a função de probabilidade acumulada da distribuição *Normal* padrão.

A.4 Teste de Entropia Aproximada

Testes de Entropia Aproximada são baseados em padrões de repetição de "strings", ou sequência de caracteres. Se

$Y_i(m) = (\epsilon_i, ..., \epsilon_{i+m-1})$, então

$$C_i^m = \frac{1}{n+1-m}\#j : 1 \le j < n-m, Y_j(m) = Y_i(m) = \pi_l \tag{A.10}$$

e por consequência

$$\Phi^m = \frac{1}{n+1-m} \sum_{i+1}^{n+1-m} \log C_i^m, \tag{A.11}$$

C_i^m é a frequência relativa de ocorrências do padrão $Y_i(m)$ na sequência analisada, e $-\Phi^m$ é a *Entropia* da distribuição que, empiricamente, surge no conjunto de todos os 2^m padrões possíveis de padrões com comprimento de bit m.

$$\Phi^m = \sum_{l=1}^{2^m} \pi_l \log \pi_l, \tag{A.12}$$

Onde π_l é a frequência relativa do padrão $l = (i_l, ..., i_m)$ na sequência.
A *Entropia* Aproximada $ApEn$ de ordem m, $m \geq l$ é definida como

$$ApEn(m) = \Phi^m - \Phi^{m+1} \tag{A.13}$$

Com $ApEn(0) = \Phi^1$.

Apêndice B

Colisões em programas de Hash

Aqui será feita a demonstração matemática da possibilidade de colisões em algoritmos de *hash*.Inicialmente algumas definições.

- q: número de saídas observadas do algoritmo com função *hash*.
- H: função *hash* executada por um algoritmo.
- N: número de diferentes sequências binárias de tamanho ℓ pertencentes ao espaço $\{0,1\}^{\ell}$ uniformemente distribuídas.
- x Entrada do algoritmo H. Assume-se que todos os valores $x_1, ..., x_q$ são distintos.
- y Saída da função H, tal que $y_i = H(x_i)$. Cada saída y_i corresponde a uma entrada x_i, tal que $y_1, ..., y_q$.
- i, j: índices inteiros compreendidos no intervalo $1 \leq i, j \leq q$.
- $coll(q, N)$: função cuja saída é a probabilidade de existirem pelo menos $y_i = y_j$, dado que $i \neq j$.
- $Pr[NoColl_q]$: Probabilidade de que não existam valores iguais dentro das q amostras de saída do algoritmo H. Ou seja, $y_j \neq y_k$, para todo $j < k \leq i$.

A questão a se verificar é se, para q amostras de y, qual a probabilidade de que $y_i = y_j$, para $i \neq j$. Primeiro, a firmação e, sem seguida, a demonstração matemática.

Seja um valor fixo de $N = 2^{\ell}$, e assumindo que $q \leq \sqrt{2N}$ amostras $y_1, ..., y_q$ são escolhidas de forma aleatória, uniforme e independente do conjunto de sequências binárias N. Então a probabilidade que existam pelo menos dois valores distintos de i, j, consequentemente $y_i = y_j$, pode ser calculada por $\frac{q(q-1)}{4N}$.

Prova: Se $NoColl_q$ ocorre então $NoColl_i$ deve também ocorrer para todo $1 \leq i \leq q$. Então,

$$Pr[NoColl_q] = Pr[NoColl_1].Pr[NoColl_2|NoColl_1]...Pr[NoColl_q|NoColl_{q-1}]$$

Tem-se que, se $Pr[NoColl_1] = 1$ uma vez que y_1 não pode colidir com ele mesmo. Além disso, se o evento $NoColl_i$ ocorre, então o intervalo gerado $\{y_1, ..., y_i\}$ contém i distintos valores. Então, a probabilidade que y_{i+1} colida com um destes valores é $\frac{i}{N}$ e, portanto, a probabilidade que y_{i+1} **não colida** com qualquer destes valores é $1 - \frac{i}{N}$. Isso significa

$$Pr[NoColl_{i+1}|NoColl_i] = 1 - \frac{i}{N},$$

então

$$Pr[NoColl_q] = \prod_{i=1}^{q-1}(1 - \frac{i}{N}).$$

Neste ponto, cabe uma adaptação da série infinita que define e^{-x}.

$$e^{-x} = 1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} + ... \tag{B.1}$$

Se x torna-se muito pequeno, então os termos além de $1 - x$ serão muito pequenos e tendem a uma insignificância matemática. Consequentemente, para valores pequenos de x pode-se aproximar o valor de e^{-x} a $1 - x$, a partir B.1 da seguinte forma:

$$1 - x \leq e^{-x} \tag{B.2}$$

Uma vez que $\frac{i}{N} < 1$ para todo i, tem-se que $1 - \frac{i}{N} \leq e^{-i/N}$, a partir de B.2, tem-se que

$$Pr[NoColl_q] \leq \prod_{i=1}^{q-1} e^{-i/N} = e^{-\sum_{i=1}^{q-1}(i/N)} = e^{-q(q-1)/2N}. \tag{B.3}$$

Utilizando a desigualdade $e^{-x} \leq 1 - (1 - \frac{1}{e})x \leq 1 - \frac{x}{2}$ no último termo de B.3:

$$Pr[Coll_q] = 1 - Pr[NoColl_q] \geqslant 1 - e^{-q(q-1)/2N} \geqslant \frac{q(q-1)}{4N}$$

Observando que $e^{-q(q-1)/2N} < 1$.

Apêndice C

Projeto SL4A

C.1 Projeto SL4A

Conforme mencionado na seção 3.1, para a codificação do protótipo usuário da aplicação de pagamentos, foram utilizadas as bibliotecas disponíveis no projeto originalmente desenvolvido pelo Google chamado **SL4A** (*Script Layer for Android*). Este projeto tem por objetivo desenvolver bibliotecas em diversos ambientes baseados em linguagem *script*, tais como Python, Pearl, Shell Bash, etc. para manipulação das API's disponíveis no Android. Existe uma boa documentação existente mas há nítida limitação de API's. Ou seja, uma aplicação visando explorar os recursos Android disponíveis deve privilegiar a linguagem nativa do sistema: *Java*.
Entretanto, se o objetivo é a criação de um protótipo apenas para validação de conceitos, então o recurso é bastante interessante tendo em vista que as linguagem baseadas em *script* são notoriamente fáceis de manipular. Especialmente para o teste de conceito desta proposta, foram utilizados os ambientes Python e Shell, devido a facilidade que estas linguagens ofereciam ao projeto.
A figura C.1 apresenta a página do grupo SL4A. O endereço de acesso é

`https://code.google.com/p/python-for-android/`

Figura C.1: Página do grupo de desenvolvimento do projeto SL4A.

Na figura C.2, pode ser vista a interface do projeto uma vez instalada no aparelho que será a base de testes do protótipo. Na verdade, trata-se de uma aplicativo desenvolvido em *Java* que interpreta os comando dados na linguagem *script* solicitada para os comandos nativos do aparelho.

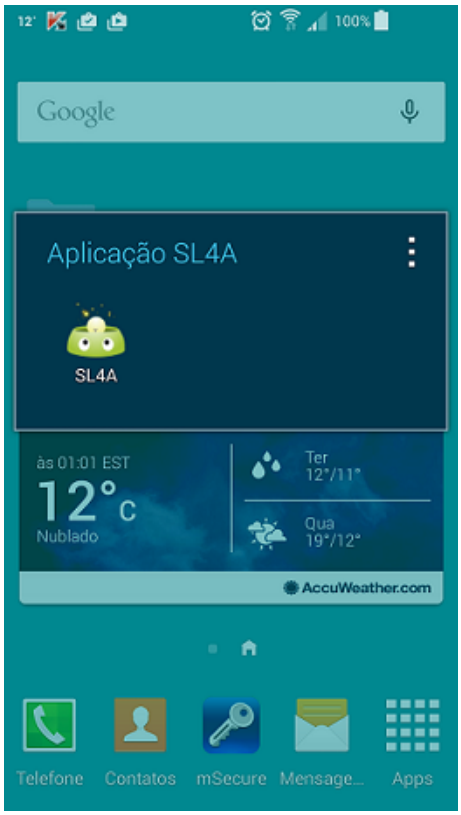


Figura C.2: Ambiente SL4A e interface de usuário.

Na figura C.3, pode ser vista a interface do aplicativo SL4A para acesso aos programas criados. Pode-se navegar utilizando o menu interativo para editar, executar e renomear o projeto. Em destaque, o projeto "$prototipo_payment_mestrado$"cuja função era receber o token gerado pelo servidor Linux, utilizando uma conexão TLS padrão e executar as funções hash.

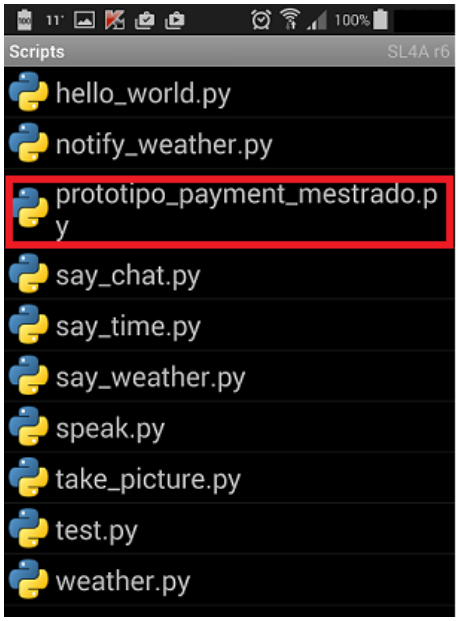


Figura C.3: Ambiente SL4A e interface de usuário.