Fernando Kaway Carvalho Ota

Autenticação de Dispositivos Móveis usando NFC

São José do Rio Preto

2016

Fernando Kaway Carvalho Ota

Autenticação de Dispositivos Móveis usando NFC

Dissertação de Mestrado

Universidade Estadual Paulista – UNESP
Instituto de Biociências, Letras e Ciências Exatas
Programa de Pós-Graduação em Ciência da Computação

Orientador: prof. Dr. Aleardo Manacero Junior

São José do Rio Preto
2016

À minha família.

Agradecimentos

Primeiramente agradeço à Deus por ter provido todos os recursos necessários, tanto intelectuais quanto materiais, para a realização desse trabalho. Agradeço minha mãe Jane por ter sempre me apoiado a correr atrás dos meus sonhos, meu pai Tadashi por me incentivar a ser uma pessoa criativa e minha irmã Larissa por ter ser minha companheira de aventuras.

Agradeço minha amada esposa Natália por ter suportado minhas loucuras e contribuído para manter o equilíbrio emocional. Meus agradecimentos também à minha nova família, Gisela, Carlos e Caio, que além de me ensinar muito sobre amor, me apoiaram na reta final do mestrado, ajudando a enfretar esse momento, que acumulou casamento e mudança para Brasília.

Agradeço especialmente meu orientador acadêmico, prof. Dr. Aleardo Manacero Jr., que acreditou em mim sem nem mesmo me conhecer, e me ensinou muito sobre liderança.

Ao grupo de pesquisa austríaco Usmile, meus sinceros agradecimentos por terem me recebido na Johannes Kepler University durante 6 meses, e terem me adicionado um conhecimento técnico essencial à realização desse trabalho.

Por fim agradeço o Banco do Brasil por ter patrocinado minha pesquisa, e os orientadores técnicos do banco Dioraci, Nelson e Zonta por terem dado um toque corporativo ao trabalho.

"Os problemas significativos que enfrentamos não podem ser resolvidos no mesmo nível de pensamento em que estávamos quando os criámos."
(Albert Einstein)

Resumo

O desenvolvimento de tecnologias móveis tem criado oportunidades para uso de aplicações remotas executando em dispositivos como smartphones. Para algumas dessas aplicações é essencial que a autenticação seja feita de modo seguro e eficiente. Nesse sentido surge o uso da tecnologia NFC (*Near Field Communication*) para obter segurança para transações executadas em aplicativos móveis. Esse tipo de aplicação é bastante interessante para a realização de comércio eletrônico, bem como controle de acesso a informações sigilosas, como dados bancários, por exemplo. Neste trabalho, apresentam-se dois protocolos para fazer a autenticação de dispositivos através de etiquetas NFC, com técnicas de criptografia assimétrica usando algoritmos de curvas elípticas. Ao longo do texto são apresentados os principais conceitos sobre NFC, criptografia e a classe de web services REST (Representational State Transfer), que serve como padrão para a construção dos protótipos dos protocolos Protecting Touch aqui desenvolvidos. Os detalhes sobre a implementação desses protocolos são também apresentados, assim como resultados de experimentos para sua avaliação.

Palavras-chaves: NFC, aplicativos móveis, segurança e criptografia.

Abstract

The appearance of mobile technologies created opportunities for remote applications running in devices such as smartphones. In some of these applications it is essential that authentication have to be performed in a secure and efficient way. In this sense the use of NFC (Near Field Communication) technology came to provide safety to transactions executed through mobile applications. This kind of application is rather attractive to perform electronic commerce, as well as access control for sensitive data, such as banking accounts. In this work it is presented two protocols for user authentication through NFC tags, using asymmetric cryptography by elliptic curves. Throughout the text we present the main concepts about NFC, cryptography and the web services REST (Representational State Transfer), which is used as the building framework for the prototypes of the Protecting Touch protocols presented here. Details about the protocols implementation are also presented, as well as results from the evaluation experiments of these protocols.

Keywords: NFC, mobile applications, security and cryptography.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

AAR	Android Application Record
API	Application Programming Interface
APDU	Application Protocol Data Unit
CRC	Cyclic Redundant Check
ECDH	Elliptic Curves Diffie-Hellman
ECMA	European Computer Manufacturers Association
EMV	Europay, MasterCard and Visa
FEBRABAN	Federação Brasileira de Bancos
HCE	Host-based card emulation
HTTP	Hypertext Transfer Protocol
IDC	International Data Corporation
IDE	Integrated Development Environment
IEC	International Eletrotechnical Comission
ISO	International Standardization Organization
JPAKE	Juggling Password Authenticated Key Exchange
JPEG	Joint Photographic Experts Group
ADS	Android Dispatch System
JSON	JavaScript Object Notation
KDF	Key Derivation Function
LLCP	Logical link control protocol
MiM	Man-in-the-Middle
NDEF	NFC Data Exchange Format
NFC	Near Field Communication

NSA	National Security Agency
POS	Point of sale
REST	Representational State Transfer
RFID	Radio Frequency Identification
RTD	Record Type Definition
SHA	Secure Hash Algorithm
SIM	Subscriber identification module
SWP	Single Wire Protocol
UICC	Universal Integrated Circuit Card
URI	Unified Resource Identifier
XML	Extreme Markup Language

Sumário

1 Introdução

Em setembro de 1991 Mark Weiser descreveu pela primeira vez o conceito de computação ubíqua para a revista Scientific American (WEISER, 1991), abordando a presença da informática em todos os lugares. A publicação já previa que os elementos especializados de hardware e software conectados evoluiriam para um estágio no qual as tecnologias empregadas seriam tão indistinguíveis que se tornariam invisíveis aos olhos da maioria das pessoas. Tal teoria se faz presente na atualidade tendo se tornado impossível para moradores urbanos saírem de casa sem ter algum contato com a computação, estando ela presente em diversas atividades, desde fazer compras no mercado até sacar dinheiro em caixas eletrônicos.

Um exemplo da ideia de Weiser são os celulares, smartphones e tablets. Esses dispositivos portáteis estão se transformando em plataformas computacionais. Celulares não são mais exclusivamente para conversar, tendo se tornado capazes de carregar vídeos e dados. Seja um telefone ou tablet, o dispositivo móvel agora é capaz de processar computação de propósito geral de forma bem semelhante a um computador pessoal. Essa tendência aumenta a gama e a participação de mercado dos aplicativos móveis (KOMATINENI; MACLEAN; HASHIMI, 2011).

O surgimento da Internet móvel através da rede de celulares permitiu que informações remotas fossem disponibilizadas a dispositivos portáteis. Aliado à popularização dos aparelhos, o acesso à rede mundial de computadores atualmente pode ser efetuado com extrema facilidade por praticamente qualquer pessoa. Em 2012, 46% dos celulares no Brasil eram smartphones (WEBXTOOL, 2012), cujas grandes diferenças em relação aos telefones celulares comuns são a possibilidade de executar aplicativos de terceiros e o acesso a Internet em melhor velocidade. Com essas facilidades é plausível o interesse das empresas em estabelecer conteúdo nesses dispositivos.

A mobilidade corporativa está mudando o dia-a-dia das operações das empresas no Brasil e algumas tendências globais já estão sendo adotadas no mercado nacional, como revela um estudo feito pela Frost Sullivan, a pedido da Motorola Solutions. O levantamento apresenta a utilização cada vez maior de soluções móveis, como o tablet empresarial ou tecnologias como a RFID (*Radio Frequency Identification*).

Adicionalmente, novas funcionalidades estão sendo incorporadas aos aparelhos celulares. Uma das mais recentes tecnologias embarcadas nos novos smartphones é o Near Field Communication, ou NFC, que permite a comunicação bidirecional sem fio entre dois dispositivos compatíveis. O NFC requer que esses dispositivos se aproximem a poucos centímetros de distância para disparar a comunicação, também conhecido como paradigma

do toque. Os dispositivos compatíveis atualmente são as etiquetas NFC, os leitores NFC e os dispositivos móveis, como os tablets e os smartphones.

As etiquetas NFC são microchips de dados compatíveis com o NFC. Uma característica interessante das etiquetas é que elas não possuem fonte de energia própria e necessitam da aproximação de um dispositivo compatível para receberem energia para o funcionamento, o que as tornam dispositivos passivos. Dessa forma sempre quem iniciará a comunicação numa interação envolvendo etiquetas NFC será um dispositivo móvel ou um leitor NFC (COSKUN; OK; OZDENIZCI, 2013b). Os dispositivos móveis também estão preparados para ler as etiquetas e disparar alguma interação com o usuário, como inicializar algum aplicativo.

Alguns exemplos da utilização da tecnologia são o bilhete único para pagamento do transporte público na cidade de São Paulo (SPTRANS, 2013), cujos leitores NFC instalado nos ônibus ou estações de metrô e trem fazem a leitura das etiquetas NFC embutidas nos cartões de bilhete único, assim como o sistema de pagamentos Google Wallet (WALLET, 2011) em funcionamento nos Estados Unidos, que permite que os usuários façam pagamentos utilizando o celular como se fosse um cartão de crédito. O banco britânico Barclay criou um serviço para que pagamentos de até 30 libras esterlinas sejam realizados simplesmente encostando uma etiqueta atuando como um elemento de segurança anexada a um celular NFC (BARCLAYS, 2012).

O setor bancário é o que mais investe em tecnologia da informação no BrasilN (FEBRABAN, 2013). Os maiores bancos nacionais já desenvolveram aplicativos móveis para que os clientes realizem transações financeiras. E para que as instituições financeiras entreguem serviços móveis fidedignos e seguros aos clientes, uma parte considerável desse investimento deve ser aplicado em pesquisas de segurança da informação.

O NFC está em plena expansão mundial. Grandes empresas adotaram a tecnologia para meios de pagamento. No Brasil o NFC vem sendo utilizado majoritariamente para controle de acesso em transporte público e estacionamento, com grande potencial de ampliação. Dessa forma, o investimento em pesquisa na área abre caminhos para invenção de novas soluções que podem ser adotadas em diversos modelos de negócios.

1.1 Motivação

As técnicas usadas nas tentativas de burlar os sistemas computacionais dos bancos são diversas. Elas vão desde a engenharia social, que compreende as práticas utilizadas para obter acesso a informações importantes ou sigilosas por meio de enganação ou exploração da confiança das pessoas, até invasões lógicas através da Internet, como ataques de hackers.

Um dos grandes problemas no mercado bancário é o roubo de credenciais dos

clientes. Os atacantes criam páginas falsas (incluindo para smartphones) com o intuito de se apoderar dos mecanismos de acesso às contas bancárias, especialmente as senhas. Uma vez em posse das credenciais, os atacantes acessam as contas através de outros dispositivos, e realizam transações espúrias. As senhas também tem sua dificuldade de memorizacão, principalmente por pessoas mais idosas.

Outra adversidade encontrada no ambiente móvel é o monitoramento de transações em dispositivos. Muitos desses dispositivos não implementam identificadores ou esses colidem. Dessa forma, há uma dificuldade em saber em qual dispositivo determinada transação fraudulenta foi executada. Como os smartphones são, em sua grande maioria, dispositivos pessoais, garantir que determinada pessoa usa um determinado dispositivo para acessar serviços contribui para a mitigação de riscos. Uma alternativa para identificar os dispositivos, muito utilizada pelos bancos, é o envio de SMS de confirmação, e salvando algum material criptográfico nesse dispositivo. Contudo, há APIs nativas de alguns sistemas operacionais que permitem a livre leitura desses SMSs. Dessa forma, um atacante poderia distribuir um *malware* em lojas virtuais, e ler as mensagens de confirmação do banco, conseguindo assim habilitar outros dispositivos. Também não há como garantir que as operadoras de telefonia móvel confirmem a autenticidade do usuário das linhas, principalmente com a existência da portabilidade de linhas de telefonia celular.

Considerando esses fatores, uma alternativa para autenticação de dispositivos e criptografia de transações móveis, que minimize os ataques se faz necessária. Em algumas situações é comum os usuários utilizarem cartões, como em bancos e transporte público. Portanto, os cartões podem figurar como um fator adicional de segurança mitigando riscos, já que um potencial atacante deveria ter posse de dois dispositivos distintos.

1.2 Objetivos

Dentro desse contexto surgiu a ideia de criar um mecanismo de segurança para aplicativos móveis distribuídos utilizando etiquetas NFC como chave criptográfica. Aproveitando o recurso de disparo de interações da tecnologia, a etiqueta, além de armazenar a chave de codificação das mensagens, também fará com que seja inicializada a aplicação no smartphone ou tablet assim que ocorrer o contato da etiqueta com o dispositivo. Nesse contato o aplicativo móvel faz a leitura da chave criptográfica armazenada na etiqueta e a utilizará para estabelecer uma comunicação confidencial e autenticada com o servidor de dados da aplicação. Esses eventos estão ilustrados na Figura 1, sendo:

1. A etiqueta NFC é aproximada ao dispositivo móvel;
2. O dispositivo móvel faz a leitura do material criptográfico e o comando para disparar a aplicação;

Figura 1 – Criptossistema do projeto

3. O aplicativo é iniciado e faz operações criptográficas com um servidor para gerar uma chave de sessão;

4. Uma sessão de troca de mensagens criptografadas e autenticadas é estabelecida.

Para implementar um mecanismo seguro com o uso de NFC é preciso atingir os seguintes resultados:

- Especificar um mecanismo de autenticação e criptografia para aplicativos em dispositivos móveis utilizando a tecnologia NFC e suas etiquetas;
- Identificar os melhores algoritmos de criptografia para o ambiente computacionalmente restrito dos smartphones e das etiquetas;

.

1.3 Estrutura do texto

A presente dissertação foi dividida em quatro capítulos, fora o introdutório. No próximo capítulo serão apresentadas a tecnologia NFC, e um embasamento sobre criptografia e *Web Services REST*, necessárias para a elaboração dos protocolos Protecting Touch (PT) e seus protótipos. No terceiro capítulo são definidos os protocolos PT, foco principal do trabalho. No quarto capítulo se descreve a implementação dos protótipos do PT, os testes realizados nesses protótipos, e uma análise dos resultados. Conclusões sobre o trabalho, assim como indicações de trabalhos futuros aparecem no último capítulo.

2 Referencial Teórico

Neste capítulo serão apresentados os conceitos básicos da tecnologia NFC de comunicação por contato, os princípios de criptografia assimétrica, para desenvolvimento da codificação das mensagens que transitarão entre o smartphone e os servidores, e também os *Web Services REST*, responsáveis por disponibilizar os conteúdos do servidor para os smartphones.

2.1 Tecnologia NFC

Philips e Sony em conjunto introduziram a tecnologia de comunicação NFC em 2002. A ECMA adotou o padrão NFC em Dezembro de 2002. A ISO e a IEC adotaram a tecnologia NFC em Dezembro de 2003. Em 2004, Nokia, Philips e Sony fundaram o Forum NFC ((NFC Forum, 2004)), uma associação de indústrias sem fins lucrativos, para promover o desenvolvimento da tecnologia NFC e seus serviços. (COSKUN; OK; OZDENIZCI, 2013a)

A tecnologia NFC utiliza a frequência de 13.56 MHz, com uma largura de banda máxima de 424 Kbps, para realizar a comunicação sem fio de curta distância. Como os dispositivos precisam se aproximar a menos de 4 cm de distância para iniciar a comunicação, essa tecnologia ficou conhecida por funcionar praticamente por toque, mas na verdade não é necessário o toque e sim uma aproximação para que a comunicação sem fio funcione. Os dispositivos compatíveis com a tecnologia NFC são:

- **Dispositivos móveis habilitados ao NFC**: os novos smartphones e tablets estão embarcando antenas NFC em seus componentes. Essa fusão de tecnologias aumentou a disseminação do NFC, ampliando largamente sua gama de aplicações.
- **Leitores NFC**: esses dispositivos foram criados especificamente para recuperar os dados provenientes das etiquetas e dos dispositivos móveis com NFC. Os mais comuns atualmente são os utilizados para controle de acesso em transporte público e de atrações de lazer, além de terminais de ponto de venda de cartão de crédito, como visto na Fig. 2a.
- **Etiquetas NFC**: são chips com memória de armazenamento que podem ser incorporados a diversos elementos, sendo mais comumente encontrados em adesivos e cartões. Não possuem fonte de energia, dessa forma para funcionar dependem do campo magnético ativado pelos leitores e dispositivos móveis NFC. Na Fig. 2b pode ser vista a comparação de tamanho da etiqueta a uma moeda.

(a) Máquina POS e smartphone NFC extraído de (HWMASTER, 2013)

(b) Etiqueta NFC e moeda

Figura 2 – Dispositivos compatíveis NFC

Nas interações entre os dispositivos NFC sempre uma das partes deve iniciar a comunicação criando um campo magnético. Os dispositivos móveis são preparados para funcionarem como iniciador ou como alvo da interação, dependendo do modo operacional que será utilizado. O leitor NFC normalmente atua como iniciador, esperando que algum dispositivo alvo o encoste. Já as etiquetas sempre serão alvo da interação, por não possuírem energia para gerar o campo magnético.

2.1.1 Arquitetura de um dispositivo ativo compatível

Para um dispositivo ser compatível com a tecnologia, ele deve conter uma antena NFC, que utiliza o protocolo de comunicação sem contato disposto na ISO 14443 (ORGANIZATION, 2003). Este protocolo tem as funções de reconhecer um dispositivo compatível e também realizar a troca de dados.

A antena deve ser integrada a um controlador projetado para habilitar as transações NFC. Nos dispositivos móveis é comum que o controlador se comunique através de um Single Wire Protocol (SWP) com um elemento seguro, de forma que suas trocas de dados sejam seguras por não transitarem por outros elementos do dispositivo.

O último elemento, dentro do dispositivo, que a tecnologia pode lançar mão é um

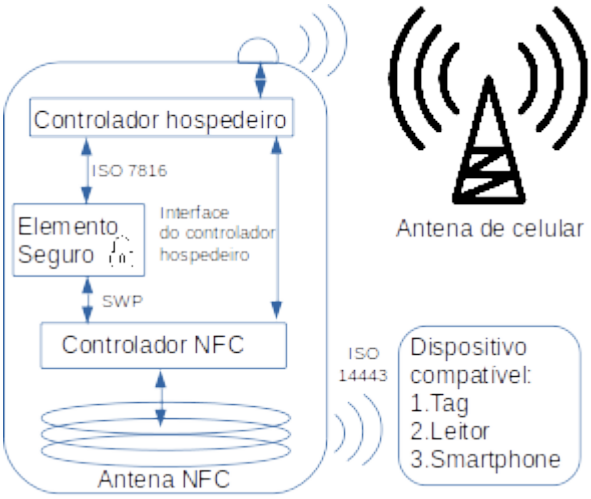


Figura 3 – Arquitetura de um smartphone NFC

controlador hospedeiro com interface de troca de dados própria caso seja necessária uma comunicação com uma rede externa, como a de celular por exemplo. Entretanto, para comunicação com o elemento seguro, o controlador hospedeiro usa o protocolo especificado pela ISO 7816.

A figura 3 apresenta um exemplo da arquitetura de um smartphone compatível com NFC. Nela é possível observar a relação entre os vários participantes do processo de comunicação, em especial as interações entre o controlador NFC e demais componentes.

2.1.2 Funcionalidades das Etiquetas NFC

O Forum NFC mantém cinco especificações para as operações das etiquetas NFC (NFC Forum, 2014a; NFC Forum, 2014b; NFC Forum, 2014c; NFC Forum, 2011; NFC Forum, 2015). Elas padronizam as estruturas de memórias e as interfaces de comunicação para acesso a diversos tipos de hardware de etiquetas NFC. A memória pode ser organizada tanto em uma estrutura linearmente endereçável quanto orientado a arquivos. A flexibilidade das especificações do Forum NFC permitiu a indústria difinir três tipos de etiquetas:

1. Armazenamento: as etiquetas atuam somente como uma memória para armazenar conteúdo. Elas possuem baixo custo, e por isso foram amplamente adotadas desde o início da tecnologia NFC. Armazenam até 8 KB. Alguns exemplos de etiquetas desse tipo são: Mifare Classic (muito utilizada no transporte público e sistemas de estacionamento) e Topaz.

2. Processáveis: uma estrutura de aplicações e arquivos é construida no processo de fabricação. Seu custo de produção é consequentemente maior, mas justificam seu uso pelo ganho de funcionalidades. Alguns modelos de transporte público ao redor

do mundo já adotam esse tipo de etiqueta para melhorar a segurança. Possuem uma capacidade de até 8 KB. Os exemplos mais conhecidos desse tipo são o Mifare DESFire e a Sony Felica.

3. Programáveis: elas possuem um sistema operacional e são capazes de receber *applets* Java. São conhecidas também como Java Cards com interface NFC. Elas foram adotadas em maior escala por bancos, uma vez que os cartões EMV já são normalmente Java Cards. O grande entrave para um maior uso dessa etiqueta é seu alto custo, embora possuam uma capacidade de armanamento que pode chegar até 64KB. O cartão SmartMX-JCOP é único produto disponível para esse tipo de etiqueta.

A maioria das etiquetas NFC possui um idenficador que varia de 4 à 10 bytes. Os identificadores são utilizados para mecanismos anti-colisão e numeração das etiquetas NFC para os leitores NFC. Geralmente, os fabricantes os atribuem de modo a criar um identificador único para cada etiqueta fabricada. Eles normalmente são armazenados no primeiro setor da memória da etiqueta, sendo gravados de forma que permitam somente a leitura. Entretanto, alguns emuladores com capadidade de emular o valor do identificador ((DUC et al., 2009)). Dessa forma, esse identificador não pode ser utilizado como único elemento de segurança.

O único mecanismo de segurança especificado no Forum NFC é a proteção contra gravação das etiquetas. Para uma grande interoperabilidade, as especificações do Forum NFC permitirem sua livre leitura por qualquer dispositivo em proximidade menor que 10 cm. Contudo, em muitas situações isso é indesejável. Um caso problemático clássico é o transporte público londrino. As etiquetas Mifare Classic 1K foram as primeiras empregadas em seu ecossistema NFC. Em 2005, os primeiros cartões de controle de acesso às catracas do metro de Londres não possuiam qualquer mecanismo de segurança. Nessa época, ainda não existiam smartphones equipados com antenas NFC. Então para clonar o conteúdo dos cartões eram necessários leitores, ainda pouco disponíveis. Com o aumento da disponibilidade desses leitores e consequentemente ataques a plataforma, a NXP, fabricante dos cartões, criou um mecanismo de criptografia proprietário denominado CRIPTO1. Em 2007, foi divulgado um estudo em (NOHL; PLOTZ, 2007) demonstrando a fraqueza do algoritmo, que confiava na obscuridade como fator de segurança. Alguns ataques foram elaborados então por criptoanalistas. A tese de mestrado publicada em (TAN, 2009) reune esses ataques. A chegada dos smartphones equipados com antenas NFC aumentou significativamente a quantidade de ataques ao sistema, forçando a NXP evoluir de cartões de armazenamento para processados e criar um novo tipo de cartão: o Mifare DESFire. Ainda sim, DES e 3DES foram o algoritmos adotados na primeira versão do DESFire, o que permite o ataque de canal divulgado em (OSWALD; PAAR, 2011) . Mais uma vez a NXP teve que alterar seu produto incluindo o algoritmo criptográfico AES. Finalmente

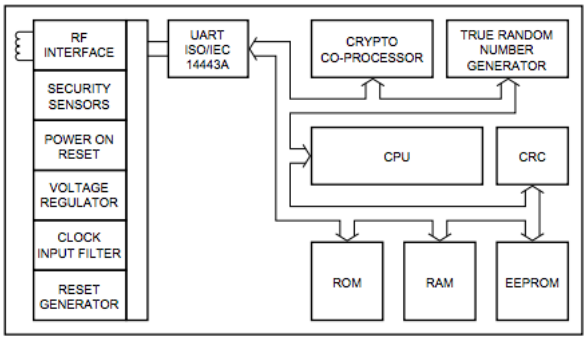


Figura 4 – Arquitetura do Mifare DESFire extraída de (NXP, 2015)

hoje a versão considerada segura é a DESFire EV1, que implementa a autenticação de três vias, gerando uma chave AES para as futuras trocas de dados criptografadas.

2.1.3 Mifare DESFire

A DESFire foi uma das primeiras etiquetas NFC a implementar um sistema autenticação de três vias com base em números aleatórios, em conjunto com um sistema de arquivos criptografados com a chave de sessão gerada pelo mecanismo de autenticação. Na arquitetura do DESFire, ilustrada na Fig. 4, é possível observer que além do gerador de números aleatórios, existe um circuito específico para CRC. O CRC nesse caso verifica se o conteúdo do arquivo não foi modificado durante a transmissão. O código para verificação vem concatenado com o conteúdo do arquivo, ambos são cifrados com a chave de sessão. O esquema para a leitura do arquivo está representado na Fig. 5.Ainda não foram encontrados ataques no sistema de segurança do DESFire, e portanto, pode-se afirmar que ele seria mais adequado do que o sistema dos cartões EMV, cujos mecanismos de segurança de senha foram encontradas brechas publicadas em (MURDOCH et al., 2010).

Para processar essa troca de informações com as etiquetas NFC processáveis e programáveis é utlizado o protocolo APDU especificado na ISO 7816-4. Inicialmente esse protocolo foi desenhado para *smart cards*, como os cartões de crédito com chip do padrão EMV. Os fabricantes de etiqutetas NFC apenas aproveitaram o protocolo já consolidado e largamente usado por bancos. Esse procotolo consiste na interface de envio de comandos em hexadecimal para os cartões e obtenção das respostas a esses comandos. A estrutura de um comando APDU está ilustrada na Fig. 6. Ele composto pela classe da instrução (CLA), o código da instrução (INS), os parâmetros das instruções (P1 e P2), o tamanho dos dados que vão ser enviados (LC), os dados, e o tamanho de resposta experada (LE).

Os comandos APDUs para DESFire são proprietários da empresa NXP, e estão protegidos por contrato de discrição. O autor do texto teve acesso aos documentos em intercâmbio de pesquisa na universidade austríaca Johannes Kepler. Porém, algumas versões antigas dos cartões DESFire anteriores foram disponibilizadas na Internet. Para

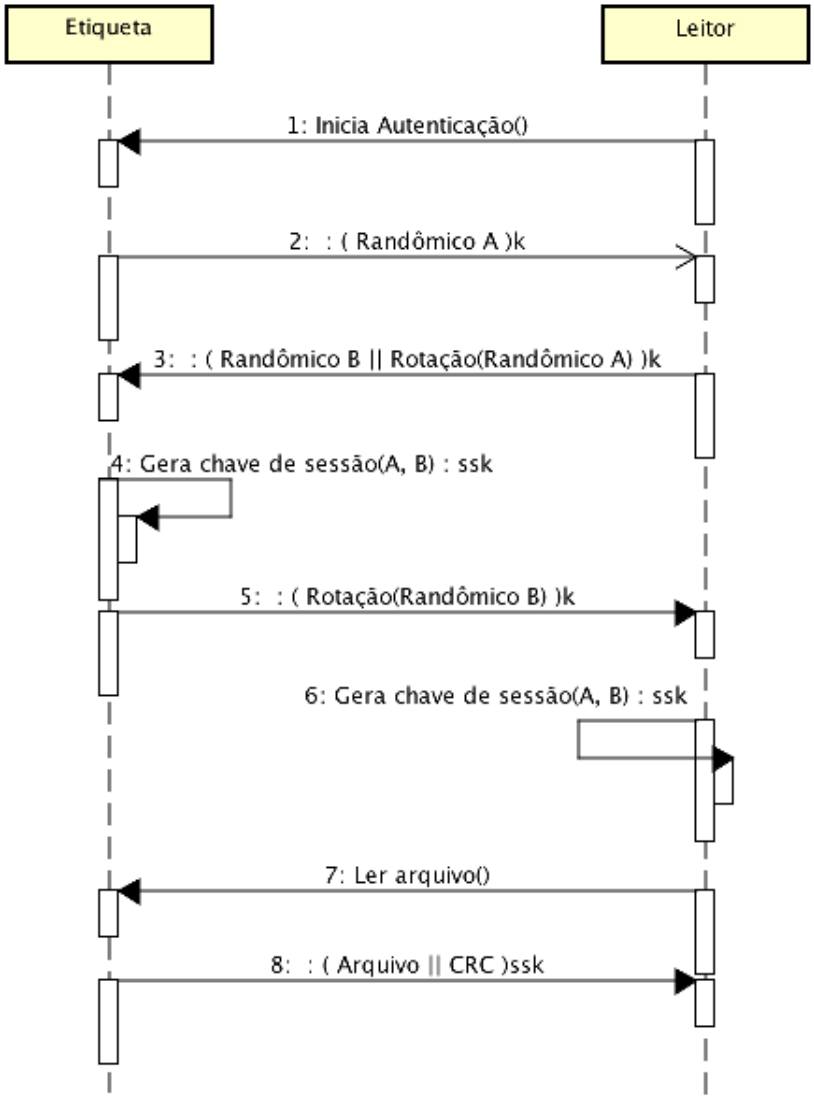


Figura 5 – Diagrama de autenticação e leitura de arquivo do Mifare DESFire

Figura 6 – Estrutura de comando APDU

```
> /send 900A0000010000
 => 90 0A 00 00 01 00 00
 (12234 usec)
 <= FB 6E 0C 61 AF 4D CC F5 91 AF
Status: 0x91AF
> /send 90AF00001071905909EC4B5893FAE427A1B23A684800
 => 90 AF 00 00 10 71 90 59 09 EC 4B 58 93 FA E4 27
    A1 B2 3A 68 48 00
 (14496 usec)
 <= F2 D1 21 A8 9C 12 1B 96 91 00
Status: 0x9100
```

Figura 7 – Exemplo de troca de comandos APDUs entre leitor e DESFire

exemplificar uma autenticação DES (disponível publicamente), a Fig. apresenta a troca de comandos do leitor OMNIKEY 5321v2 com o DESFire EV1 usando o programa JC Shell da NXP. No programa o envio de comandos é feito por meio de *send*, e a resposta é denotada por =>. O primeiro comando enviado no exemplo da figura é a para iniciar a autenticação, sendo 90 a classe dos comandos DESFire, 0A o comando de iniciar a autenticação DES de três vias, 00 para P1 e 00 para P2, que é padrão do DESFire, o campo LC tem valor 01 pois será enviado somente 1 byte, 00 no campo de dados para autenticação com a chave master do DESFire, e 00 no campo LE, também padrão do DESfire. Os 8 primeiros bytes da primeira resposta formam o número randômico criado pelo cartão e cifrado com a chave mestra, o penúltimo byte, 91, indica que o comando foi executado com sucesso, e o último byte, AF, indica que o cartão está esperando o envio de dados do próximo comando. Na sequência é enviado o comando AF para a continuação da autenticação junto com 16 bytes (10 em hexadecimal no campo LC), que é um número criado pelo usuário, concatenado com o número enviado pelo cartão rotacionado em 1 byte, ambos cifrado com a chave master. A última resposta completa a autenticação enviando 8 bytes, que é o número recebido do leitor rotacionado em 1 byte e cifrado pela chave master. Após esses comando é possível fazer modificações no cartão, como instalação de aplicativos e arquivos, bem como criação de chaves adicionais.

A última questão a ser analisada em relação às etiquetas envolve os dois protocolos de leitura: NFC-A e NFC-B. A grande diferença entre eles é o tipo de codificação de frequências. O NFC-A utiliza a codificação de Miller e o NFC-B a de Manchester. Na prática, isso reflete na questão de interoperabilidade das etiquetas com os smartphones, pois nem todos implementam os dois protocolos. Os passaportes do novo padrão internacional, que contém os dados, incluindo biometrias de face e digital, dos viajantes, são gravados em etiquetas com o protocolo NFC-B. Com isso, diversos smartphones NFC não conseguem ler essas etiquetas, inviabilizando protocolos que poderiam ser implementados usando esses dados biométricos.

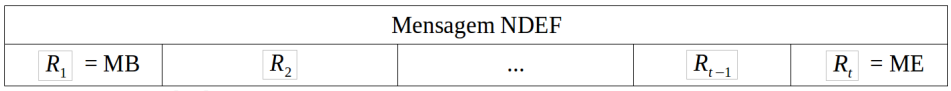


Figura 8 – Mensagem NDEF

2.1.4 Formato de troca de dados (NDEF)

O Forum NFC especifica um formato de troca de dados (NDEF) comum para os dispositivos e etiquetas NFC. A especificação define as regras de como os dados devem ser encapsulados para que qualquer dispositivo compatível possa processar as mensagens. O NDEF provê uma estrutura de dados simples e eficiente que permite acomodar:

1. Encapsulamento de documentos e entidades, inclusive criptografadas, no formato XML, bem como imagens nos formatos GIF e JPEG, etc;
2. Encapsulamento de documentos e entidade de tamanhos inicialmente desconhecidos. Essa capacidade permite que documentos grandes sejas divididos em pedaços;
3. Documentos associados em uma única mensagem;
4. Encapsulamento de pequenos *payloads* sem aumentar a complexidade dos parsers.

Uma mensagem NDEF é composta de um ou mais registros. Cada registro é uma unidade para carregar uma carga e contém seu próprio conjunto de parâmetros. O primeiro registro sempre terá uma marcação de flag denominada MB (message begin). E o último registro terá a marcada a flag ME (message end). Esses registros iniciais e finais não poderão carregar nada além das flags de marcação. Não existem limites lógicos para a quantidade de registros, ou seja, os limites são normalmente de quantidade de memória disponível.

A figura 8 apresenta o formato da mensagem que é sempre da esquerda para direita. Os registros não possuem índice, e portanto sua ordem é dada implicitamente pela forma com que são serializados.

Uma mensagem NDEF possui três parâmetros descritivos:

1. **Comprimento**: independente das relações entre os registros, o comprimento de carga indica seu tamanho encapsulado nesse registro. O campo PAYLOAD LENGTH é de um octeto para registro curtos e quatro octetos para registros normais. Registro curtos são indicados pela *flag* SR (short records);
2. **Tipo**: indica o padrão de dados carregado na carga. Esse parâmetro deve indicar o tipo de dados para toda a mensagem NDEF. O valor do tipo é indicado pelo campo

Tabela 1 – Tipos para mensagens NDEF padronizados pelo Forum NFC

Nome do tipo do formato	Hexadecimal
Vazio	0x00
URI RTD	0x01
Tipo de mídia definido no RFC 2046 [RFC 2046]	0x02
URI absoluta definida no RFC 3986 [RFC 3986]	0x03
Tipo externo RTD	0x04
Desconhecido	0x05
Não modificado (quando carregando pedaços de registros)	0x06
Reservado para uso futuro	0x07

TNF (type name format). Os valores padronizados pelo Forum NFC estão dispostos na tabela 1

3. **Identificador**: é um parâmetro opcional que permite que a aplicação identifique a carga que está sendo transportada. Para inclusão do identificador a *flag* IL deverá ser marcada e o campo ID LENGTH indicará o comprimento do identificador.

Uma mensagem pode ser maior que o tamanho disponível no dispositivo compatível (ex.: etiqueta de 1 Kb e mensagem de 3 Kb). Nesse caso é possível particionar a mensagem em vários registros. Dessa forma a flag ME será marcada somente no último registro contendo o último pedaço da mensagem, e os registros intermediários não terão ME marcados. A flag CF (chunk flag) é marcada em todos os registros para indicar que se trata de uma mensagem dividida, exceto no último. O campo identificador ID deve conter o mesmo valor para todos os registros da mensagem. A figura 9 a seguir apresenta o leiaute de um registro.

Os casos de uso dos registros NFC mais conhecidos foram padronizados pelo Forum NFC, sendo conhecidos como *Record Type Definition* (RTD). Se o campo TNF indicar o valor 0x01 que corresponde a URI RTD então, o tipo de dado transportado já está padronizado pelo Forum NFC. Essa padronização no caso das URIs consiste em determinar o par (valor hexadecimal, URI). Alguns exemplos de prefixos URI já conhecidos, que irão compor o campo TYPE do registro, são apresentandos na tabela 2

2.1.5 Modos operacionais

Existem atualmente três tipos de modos de operações NFC, que se diferenciam na maneira como são realizadas as interações entre os dispositivos, introduzidos nesta seção.

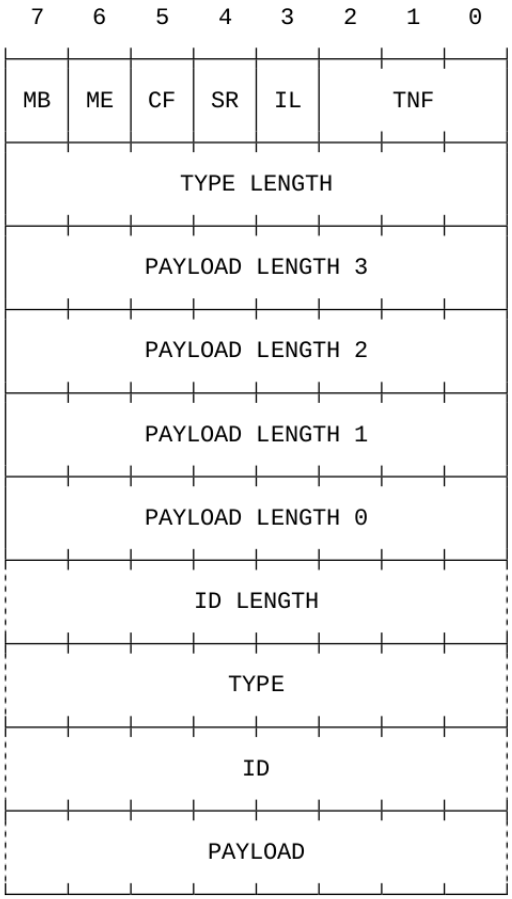


Figura 9 – Leiaute de registro NDEF (NFC FORUM, 2006)

Tabela 2 – Prefixos de URIs comuns

Prefixo	Hexadecimal
http://	0x03
tel:	0x05
mailto:	0x06
ftp://	0x0D
telnet://	0x010
imap:	0x011
Reservado para uso futuro	0x24...0xFF

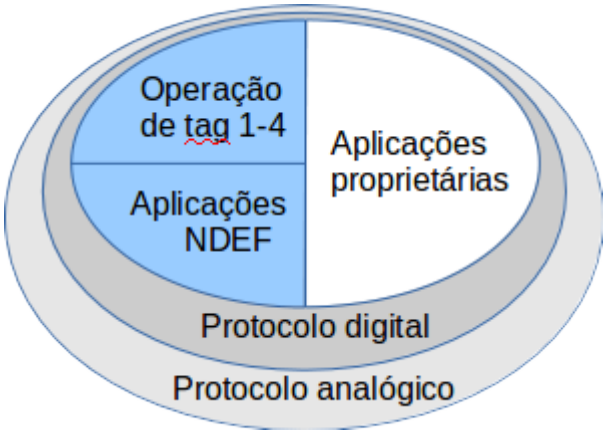


Figura 10 – Camadas do modo operacional de leitura e escrita

Leitura e gravação

No modo de operação de leitura e gravação o dispositivo móvel ou leitor NFC atua como iniciador da interação que permite leitura dos dados gravados nas etiquetas. O dispositivo móvel também pode gravar dados nas etiquetas. A única taxa de transferência de dados possível nesse modo é de 106 Kbps. A arquitetura do modo de leitura e gravação compreende os seguintes elementos e camadas, como vistos na figura 10:

- Protocolos digitais e analógicos: perfazem a comunicação de baixo nível entre os dispositivos;
- Operação de etiqueta tipo 1-4: comanda um dos quatro tipos de operações baseadas na escolha das etiquetas;
- Aplicações NDEF: realizam operações com o padrão de dados NDEF;
- Aplicações proprietárias: parte destinada a aplicações específicas fora do padrão NDEF.

Peer-to-peer

Nesse modo dois dispositivos móveis se encostam e podem transferir dados um para o outro. Apesar de bidirecional, somente uma das partes pode enviar dados por vez. Este modo permite taxas de transferência na ordem de 106, 212 e 424 Kbps, atingindo assim a taxa máxima disponível atualmente.

Esse modo também está organizado em camadas, Figura 11, sendo que utiliza o protocolo de controle de ligação lógico LLCP, baseado no padrão industrial IEEE 802.2. O fluxo de dados possível nesse protocolo é menor se comparado a outros protocolos de rede como o TCP/IP, entretanto fornece um ambiente de serviços mais robusto [NFC Forum]. Dessa forma uma camada de comunicação de dados é adicionada as transações para poder prover os serviços.

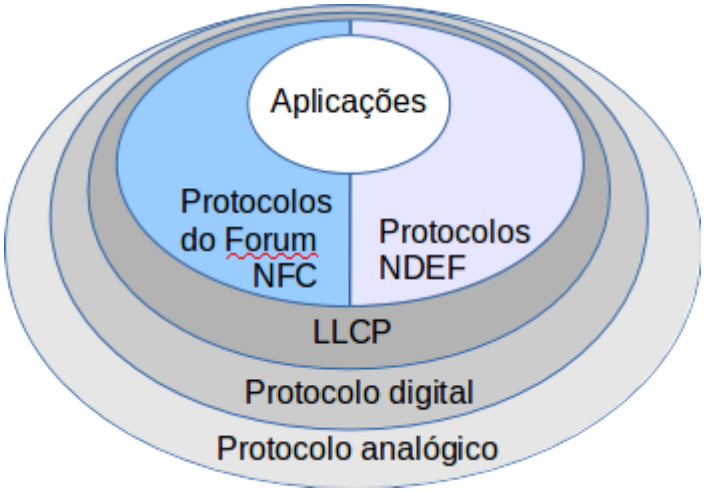


Figura 11 – Camadas do modo operacional Peer-to-peer

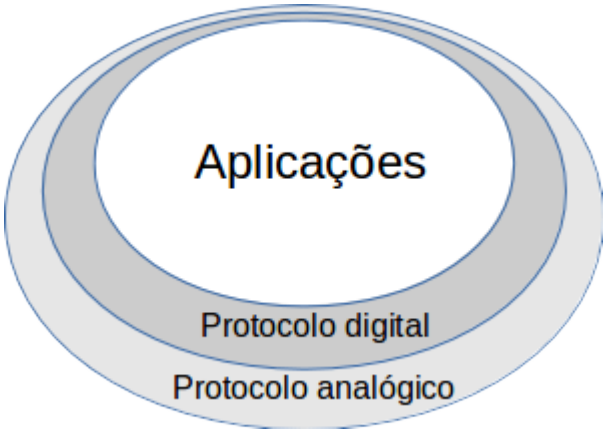


Figura 12 – Camadas do modo operacional de emulação de cartão

Emulação de cartão

Esse importante modo de operação habilita as aplicações de pagamento nos dispositivos móveis, pois permite que esses dispositivos funcionem como cartões inteligentes. Nesse caso o leitor NFC gera o campo magnético que inicia a interação, enquanto o dispositivo móvel provê os dados necessários para as transações, atuando como um cartão inteligente. A taxa máxima de transferência de 424 Kbps é a utilizada nesse modo.

A estrutura de camadas deste modo é mais simples, Fig. 12, mantando apenas os protocolos analógicos e digitais. Dessa forma o tratamento de pacotes é deixado para as aplicações, com um ganho de flexibilidade do processo.

2.1.6 Elemento Seguro

Algumas aplicações requerem que suas transações sejam feitas em ambientes com segurança. Para isso, os fabricantes de smartphones tem investido no desenvolvimento de

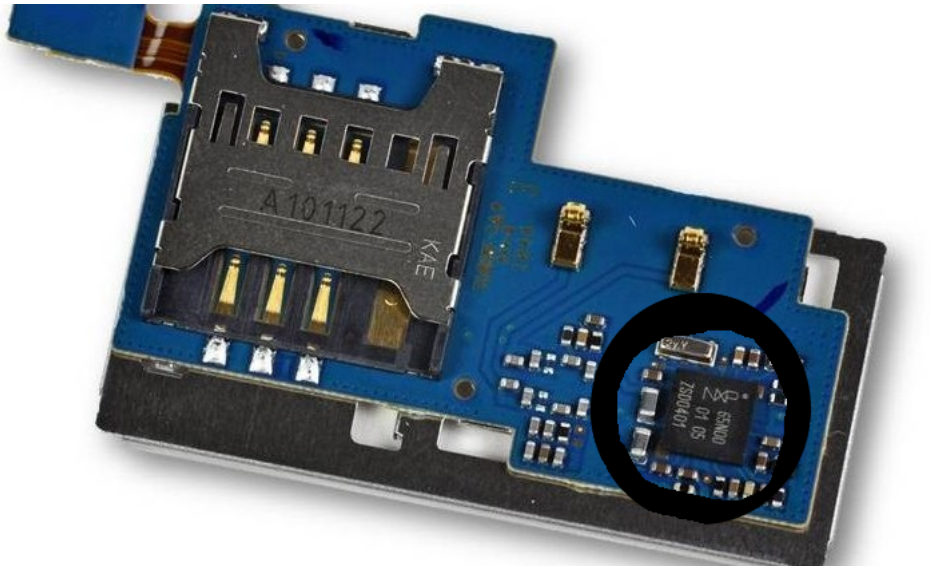


Figura 13 – Elemento seguro em hardware embaracado

elementos seguros. Esses elementos são uma combinação de hardware, software, interfaces e protocolos embarcados nos dispositivos móveis, proporcionando mecanismos de proteção ao ambiente transacional dos aparelhos.

Por se tratarem de uma combinação computacional, os elementos de segurança dependem de um sistema operacional. Esse sistema operacional deve ser capaz de coordenar todos os elementos, possibilitando a execução segura das aplicações e também o armazenamento de dados em sigilo.

Os dados armazenados em transações com dispositivos móveis não devem ser guardados na sua memória, prevenindo manipulações indevidas ou exclusão sem intenção dos dados. Uma grande variedade de elementos de segurança pode ser empregada para o armazenamento de dados sigilosos. Uma visão geral dos elementos de segurança mais comuns é apresentada a seguir.

Hardware embarcado

Um dispositivo móvel pode ser construído com um hardware para atuar como o elemento seguro. Nesse caso um chip específico para segurança é instalado no processo de fabricação do smartphone, exemplo na Fig. 13. Normalmente é necessário que seja personalizado cada vez que uma nova pessoa utiliza o aparelho. Esse tipo de dificuldade levou o Google a deixar de utilizar esse tipo de dispositivo na sua solução de pagamentos Google Wallet a partir da versão 4.4 “Kitkat” do sistema operacional Android.

Adesivos

Os adesivos são normalmente cartões inteligentes ou etiquetas desenvolvidas para serem afixadas na parte traseira de smartphones. Assim como as etiquetas NFC, os adesivos

Figura 14 – Elemento seguro em adesivo

Figura 15 – Chip GSM como elemento seguro

podem ser ativos ou passivos. Quando ativos, os adesivos se comunicam com o ambiente de processamento da aplicação fazendo parte da lógica envolvida na transação, e normalmente possuem os três modos de operação da tecnologia NFC. Se o adesivo for passivo, somente fazem operações estáticas, funcionando muito mais como dispositivos de armazenamentos de alguns dados para as transações.

Cartão de Circuito Integrado Universal (UICC)

O *Universal Integrated Card Circuit* (UICC) mais disseminado é o chip de módulo de identificação de assinante SIM, popularmente conhecido como chip do celular. Funciona como um pequeno cartão inteligente que contém uma área de armazenamento segura, tornando-o uma excelente opção de elemento seguro. Entretanto, o controle desse chip normalmente pertence às operadoras de telefonia móvel, e por isso tem sido evitado por deixar as aplicações vulneráveis as operadoras, além de onerar os desenvolvedores que terão de negociar seu uso.

2.2 Criptografia

Esta seção apresenta alguns conceitos de criptografia. Nas próximas seções algoritmos criptográficos serão combinados a fim de alcançar um nível elevado de confidencialidade

e integridade para um aplicativo móvel. Não é a ambição deste trabalho cobrir todo o conhecimento de criptografia, mas sim apresentar os algoritmos necessários para tornar seguras as arquiteturas de sistemas propostas no próximo capítulo.

2.2.1 Criptografia simétrica

Um esquema de criptografia é classificado como simétrico quando uma chave secreta compartilhada mútua é usada para cifrar e decifrar a mensagem. Esse esquema tem sido amplamente utilizado desde a primeira técnica, conhecida de cifra de César. A segurança dos algoritmos de criptografia simétrica conta com a impraticabilidade de decifrar uma mensagem secreta. Por exemplo, se um invasor tenta com força bruta decifrar a mensagem criptografada, em outras palavras, testar todas as possíveis chaves usando o poder computacional atual, o texto simples só será revelado em um tempo tão longo que tornaria o ataque inviável.

Existem dois tipos principais de algoritmos de criptografia simétrica: cifras de fluxo e cifras de bloco. Os algoritmos de cifra de fluxo consistem em criptografar cada bit do texto simples com um fluxo pseudo-aleatório. Esse fluxo é gerado por uma função que aumenta o tamanho de uma semente, a fim de gerar uma cifra com o mesmo comprimento da mensagem. Em seguida, o texto simples é combinado com este fluxo de criar o texto cifrado. Este tipo é muito utilizado para proteger conexões remotas sem fio, como redes de telefonia móvel. A sua implementação depende de outros mecanismos para fornecer proteção de integridade ou autenticação.

A criptografia por cifra de bloco é a melhor escolha para criptografar as mensagens sensíveis entre o servidor e o smartphone, por não depender de um canal com estado, sendo adequado ao REST. Este tipo de algoritmo utiliza uma chave de comprimento fixo para cifrar blocos de uma mensagem.

Vários estudos analisaram o estado da arte da criptografia por cifra de blocos, com destaque para (MATHUR; KESARWANI, 2013) .Os principais algoritmos não-patenteados foram investigados neste estudo, sendo recomendado o Advanced Encryption Standard (AES), pois é a única opção segura e gratuita dentre os avaliados. Este algoritmo ganhou um concurso criado pelo Instituto Nacional Americano de Padrões e Tecnologia (NIST), a fim de substituir o Data Encryption Standard (DES), e sua descrição pode ser encontrada em (DAEMEN; RIJMEN, 1998).

Outro ponto a ser considerado é o desempenho devido às restrições dos smartphones e das redes de internet móvel. Este último aspecto foi avaliado em (ABDUL; ELMINAAM; HADHOUD, 2009) apontando que Blowfish é o algoritmo com melhor desempenho, seguido pelo RC6, e pelo AES. Como o estudo anterior classificou os primeiros dois como vulneráveis, o AES fica como a melhor opção. Em (ABDUL; ELMINAAM; HADHOUD, 2009) também

se avalia o efeito de mudar o tamanho das chaves AES, monstrando que o aumento do tamanho de chave de 128 bits (o tamanho mínimo para AES) para 256 bits implica em um aumento de 16% no consumo de energia e tempo de desempenho.

Se o AES for empregado individualmente, a saída da cifragem de duas mensagens idênticas seriam iguais. Em vez disso, é possível aplicar Modos para embaralhar essas saídas de forma que um intruso não reconheceria a troca de textos idênticos. Uma avaliação dos principais algoritmos desses modos foi realizada para o Governo japonês, e publicado em (ROGAWAY, 2011). Os autores dividiram os modos de cifragem em três tipos:

- Modos de Confidencialidade: este modo faz permutações da mensagem original (em um texto cifrado no caso) para garantir a privacidade. Entre os 6 métodos comparados, seria possível usar o Counter Mode (CTR), uma vez que é considerado o melhor e mais moderno neste modo. O CTR requer um vetor de inicialização (IV), que não pode ser reutilizado.

- Modos de Autenticidade: esses modos são geralmente empregados quando é necessário para assinar mensagens, ou seja, criar um código de autenticação exclusivo para cada entrada diferente. A mensagem Código de Autenticação de hash (HMAC) é apontada como o mais eficiente no estudo, e é a escolha para proteger os protocolos de troca de chaves usados neste trabalho.

- Modos item de criptografia autenticado: combinam confidencialidade e propriedades de autenticidade dos outros modos. Os dois únicos algoritmos de comparação neste modo são Counter Mode com CBC-MAC (CCM) e Galois Counter Mode (GCM). Ambos têm segurança, no entanto GCM é paralelizável levando a uma melhor recomendação uma vez que os smartphones tendem a ter mais núcleos de processamento.

2.2.2 Criptografia assimétrica

A criptografia assimétrica consiste em criptografar e descriptografar usando chaves diferentes – uma pública e uma privada. Ela também é conhecida como criptografia de chave pública. Ela transforma o texto claro em texto cifrado usando a chave pública e um algoritmo de criptografia. Usando a chave privada associada e um algoritmo de decriptografia, o texto claro é recuperado a partir do texto cifrado. (STALLINGS, 2008)

Para que o sistema funcione, cada usuário deve gerar um par de chaves, uma pública e outra privada. O usuário que possuir uma chave pública poderá enviar mensagens cifradas, que serão decifradas somente pelo usuário que tiver a chave privada correspondente. A Figura 16 representa esse esquema.

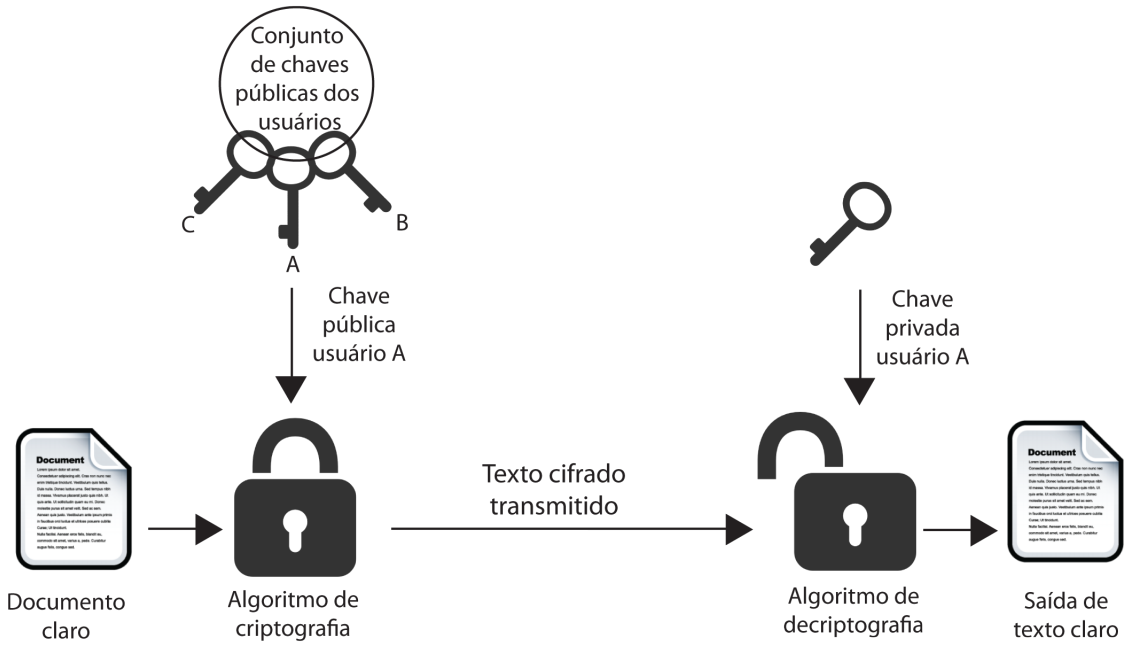


Figura 16 – Modelo de criptografia assimétrica adaptado de (STALLINGS, 2008)

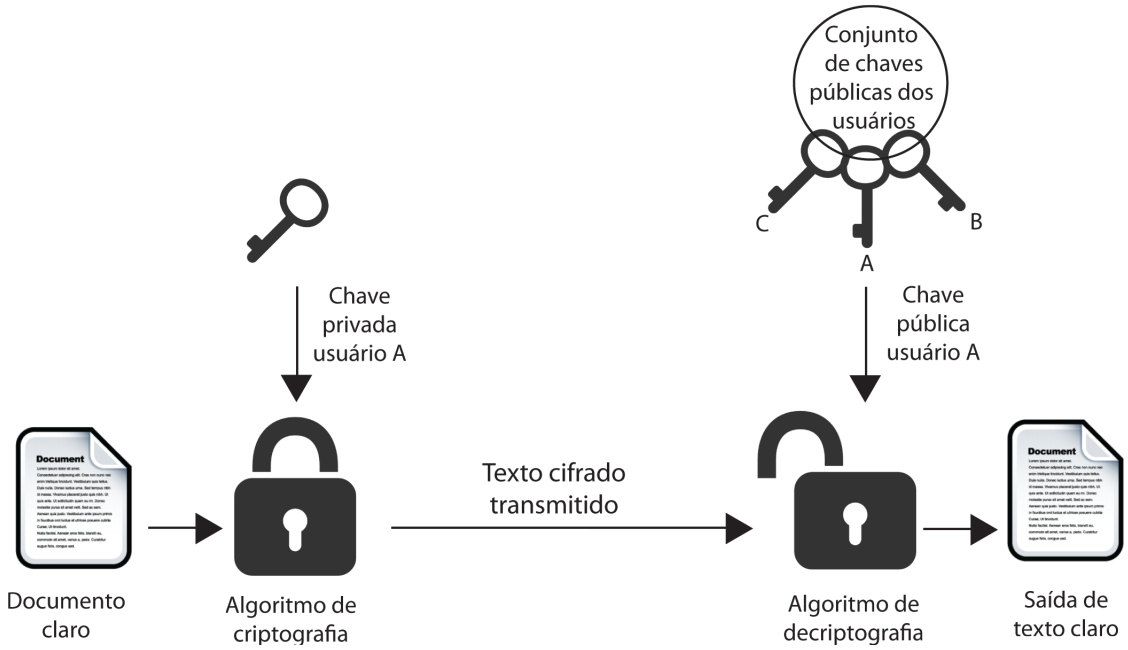


Figura 17 – Modelo de autenticação adaptado de (STALLINGS, 2008)

O usuário que gerar as chaves deve manter em segredo sua chave privada. Toda mensagem que for criptografada com essa chave terá a garantia que os usuários que usarem sua chave pública saberão que estão recebendo a mensagem do determinado remetente, essa operação também é conhecida como assinatura. A Figura 17 ilustra o mecanismo de assinatura.

Os dois algoritmos mais utilizados para os criptossistemas de chaves públicas

atualmente são o RSA e a aritmética de curvas elípticas. A segurança do algoritmo RSA está no problema de fatorar grandes números primos. Na aritmética de curvas elípticas, para poder quebrar o sistema, deve-se resolver o problema do logaritmo discreto elíptico.

Um dos atuais focos de estudo de empresas que elaboram algoritmos de segurança é a escolha do algoritmo criptográfico a empregar nos sistemas para dispositivos móveis, devido as suas limitações. As duas principais restrições dos smartphones atuais a serem consideradas para desenvolvimento de criptossistemas são:

1. Limitação de processamento: mesmo com o avanço dos processadores dos dispositivos, ainda a capacidade de processamento é inferior ao de um dispositivo não portátil;

2. Consumo de energia: como os smartphones contam com baterias finitas, o consumo de energia deve ser levado em consideração para construção de aplicativos.

Os aplicativos que necessitam de um alto processamento devem ser otimizados se o objetivo é usá-los em dispositivos móveis, pois eles afetam as duas restrições acima mencionadas ao mesmo tempo, uma vez que a criptografia é potencialmente custosa computacionalmente pois faz uso de operações matemáticas complexas. A agência nacional de segurança americana publicou em (NSA, 2009) uma avaliação dos algoritmos de criptografia recomendando o uso das curvas elípticas.

A empresa canadense Certicom, responsável por desenvolver soluções de segurança que protegem conteúdo de milhões de dispositivos no mundo, inclusive os da NSA, está conduzindo atualmente uma pesquisa sobre assinaturas e certificados digitais utilizando os algoritmos de curvas elípticas para registros de assinaturas digitais usando NFC. O estudo publicado em (ROSATI; ZAVERUCHA, 2011) propõe o uso do algoritmo de curvas elípticas para criar assinaturas digitais utilizando etiquetas NFC. Para tal foram comparados os principais algoritmos de criptografia demonstrando que os mais indicados são os que utilizam a aritmética de curvas elípticas pois utilizam chaves menores se comparadas aos métodos como o RSA, diminuindo o uso de memória. A redução da necessidade de memória é outro fator significativo para a escolha do algoritmo de criptografia, pois as etiquetas NFC com menor capacidade de armazenamento são consideravelmente mais baratas e a economia de memória chega ser de até 90 %. Entretanto a pesquisa demonstrou que o desempenho computacional não é afetado pela escolha do algoritmo uma vez que os processadores encontrados nos smartphones atuais conseguem executar os procedimentos de assinatura e verificação em menos de 10ms usando chaves de 384 bits. Os custos das etiquetas também foram quantificados no estudo. As etiquetas mais econômicas com memória de até 2048 bytes tem um preço menor que um dólar. Essa quantidade de memória seria suficiente para armazenar o caminho do aplicativo móvel e também a chave criptográfica.

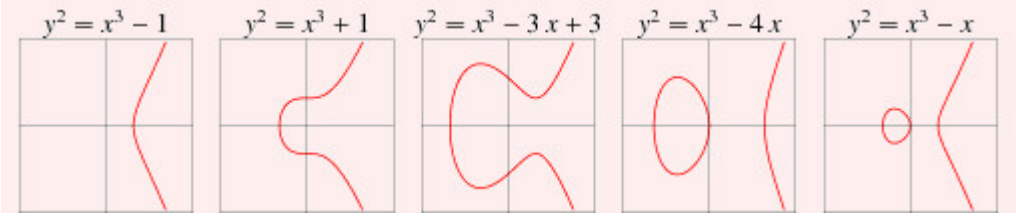


Figura 18 – Exemplos de curvas elípticas

A empresa de semicondutores neerlandesa NXP criou um algoritmo de autenticação e cifragem para ser utilizado nas etiquetas NFC Mifare Classic. Entretanto esse algoritmo foi quebrado pelos pesquisadores da Universidade de Virginia e demonstrado em (NOHL et al., 2008).

A criptografia de curvas elípticas tem também a maior força por bit comparado aos outros criptossistemas de chaves públicas, ou seja, chaves menores conseguem prover a mesma segurança (CHOU, 2003). E por isso as ECC são a melhor escolha para os ambientes computacionalmente restritos como os smartphones. Além de contribuir para o menor gasto de bateria e processamento do dispositivo, também será necessária menor largura de banda para envio dos pacotes, quando empregado esse tipo de criptografia. Mesmo com a melhora da rede de Internet móvel, essa economia reflete em menor custo para o usuário e também uma transferência mais rápida de pacotes.

As curvas elípticas foram propostas para uso em criptografia de forma independente por dois pesquisadores, em (MILLER, 1986) e (KOBLITZ, 1987). Através de operações matemáticas das propriedades das curvas elípticas é possível realizar assinaturas digitais, troca de chaves e encriptação.

O foco desse estudo é utilizar as curvas elípticas para geração e troca de chaves criptográficas usando o protocolo de Diffie-Hellman. A cifragem dos dados será feita usando algoritmos simétricos, pois são computacionalmente menos dispendiosos, porém com a combinação de chaves assimétricas. Esse protocolo visa usar a robustez de segurança das curvas elípticas associado a melhor performance dos algoritmos simétricos de criptografia.

Uma curva elíptica é um conjunto de pontos que satisfazem a seguinte Equação 2.1:

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

onde $a, b \in K$, sendo K um corpo finito. A figura 18 ilustra alguns exemplos de curvas elípticas.

O próximo passo para entender as curvas elípticas é compreender as duas operações aritméticas usadas no protocolo de troca de chaves de Diffie-Hellman: adição e multiplicação

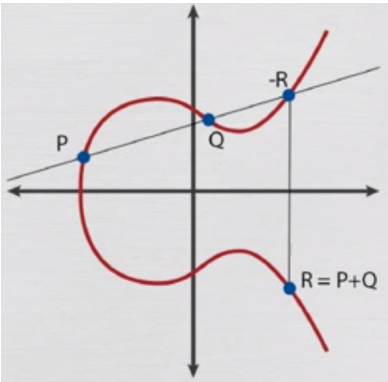


Figura 19 – Soma de pontos de curvas elípticas

por escalar. A adição de dois pontos numa curva elíptica resulta num terceiro ponto. Se traçarmos uma linha entre dois pontos, ela interceptará a curva no terceiro ponto que é o valor negativo da soma desses pontos. Então basta espelhar esse terceiro ponto para o outro lado da curva que resultará na soma dos pontos. A operação está graficamente demonstrada na Figura 19.

Algebricamente para chegar ao ponto R resultante da soma dos pontos P, Q,inicialmente é necessário achar a inclinaçãoda curva, através da Equação 2.2:

$$s = fracy_p - y_q x_p - x_q \tag{2.2}$$

A partir do valor da inclinação é possível obter as coordenadas do ponto R usando as Equações 2.3 e 2.4:

$$x_r = s^2 - (x_p + x_q), \tag{2.3}$$

$$y_r = s(x_p - x_r) - y_p. \tag{2.4}$$

Porém a adição de dois pontos verticais resulta no ponto no infinito O, conforme mostra a Figura 20.

A outra operação que permite o algoritmo de Diffie-Hellman é a multiplicação por escalar, representada pela Equação 2.5. A multiplicação de um ponto P por um escalar k resulta num ponto Q, que nada mais é que a soma de k vezes o ponto P, representada pela Equação 2.6.

$$Q = kP \tag{2.5}$$

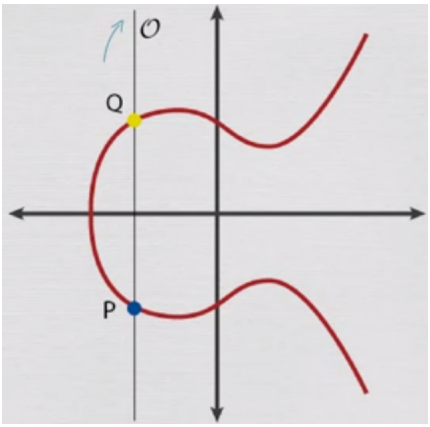


Figura 20 – Adição de pontos verticais

$$Q = P + P + P + ... + P \tag{2.6}$$

A segurança das curvas elípticas consiste na dificuldade em achar k dado o ponto Q na operação de multiplicação de um ponto P pelo escalar k, também conhecido como problema do logaritmo discreto sobre curvas elípticas. Essa função é dita de uma via, já que por meio das operações matemáticas básicas, como a divisão, não é possível chegar ao resultado de k.

A última peça faltante para o criptossistema é um ponto gerador base G, que gera um subgrupo de pontos da curva elíptica. Multiplicando esse ponto gerador G por um escalar é possível acessar todos os pontos do subgrupo. A ordem de G é o tamanho do subgrupo. Uma curva elíptica robusta para uso em criptografia é quando o número total de pontos na curva é igual o número de pontos do subgrupo gerado por G. Esse valor é chamado de cofator e é obtido pela Equação 2.7. Portanto o valor ideal do cofator é 1.

$$Cofator = \frac{numero\ de\ pontos\ da\ curva}{Ordem\ de\ G} \tag{2.7}$$

2.2.3 Troca de chaves

O algoritmo de Diffie-Hellman (DH) foi criado em 1976 por Whitfield Diffie e Martin Hellman. A segurança do algoritmo está na dificuldade em resolver problemas de logaritmos discretos como o de curvas elípticas apresentado acima.

Para realizar a comunicação privada, cada usuário cria um par de chaves assimétricas, e envia sua chave pública para os outros usuários para quem deseja enviar mensagens. Combinando sua chave privada e a chaves públicas recebidas dos outros usuário é gerada uma chave secreta.

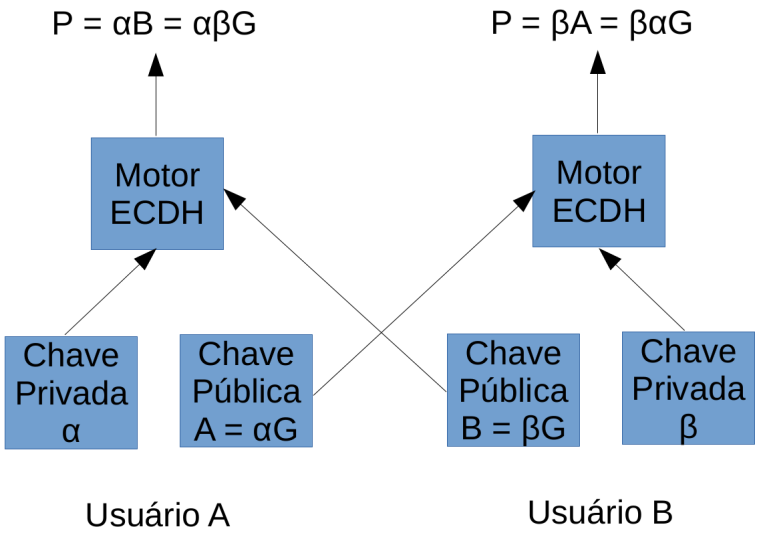


Figura 21 – Troca de chaves ECDH

Essa chave secreta é igual para todos os usuários envolvidos na mesma comunicação, e dessa forma é possível usá-la para criptografar dados usando algoritmos de criptografia simétrica. Note que também esse protocolo pode ser usado para mandar mensagens criptografadas em redes de vários usuários.

No caso de uso das curvas elípticas, há um conjunto de informações públicas que são usadas para gerar as chaves. Todas as partes devem conhecer a curva que está sendo usada, ou seja, os parâmetros a, b, bem como o ponto base gerador G, e o a ordem e o cofator. Em posse desses dados o usuário escolhe um escalar α, que será sua chave privada, e multiplica pelo ponto gerador G. O resultado dessa multiplicação será um ponto denominado A, que é a chave pública do primeiro usuário. Um outro usuário que desejar se comunicar comunicar com esse usuário, fará o mesmo processo, escolherár um escalar β, e o multiplicará pelo mesmo gerador G, resultando no ponto B. Então cada usuário multíplica o ponto recebido do outro usuário pelo seu escalar, e ambos chegarão no mesmo ponto P. Esse esquema está ilustrado na Figura 21.

Apesar do algoritmo DH ser eficiente para a geração de chaves simétricas secretas mesmo que as chaves públicas sejam amplamente conhecidas, ele é suscetível ao ataque conhecido como *Man-in-the-Middle* (MiM). Um atacante com controle do canal de comunicação pode executar trocas de chaves se passando por cada usuário, e gerando assim uma chave de comunicação com cada um deles como apresentado na Figura 22. De acordo com (HOOK, 2005), para evitar esse tipo de ataque, os participantes devem se autenticar uns com os outros. Como o DH não fornece nenhum recurso para reconhecimento de proveniência das chaves, os usuários fraudados nunca saberão que estão se comunicando com o atacante.

Existem algumas estratégias para autenticação de *hosts* implementadas na indústria.

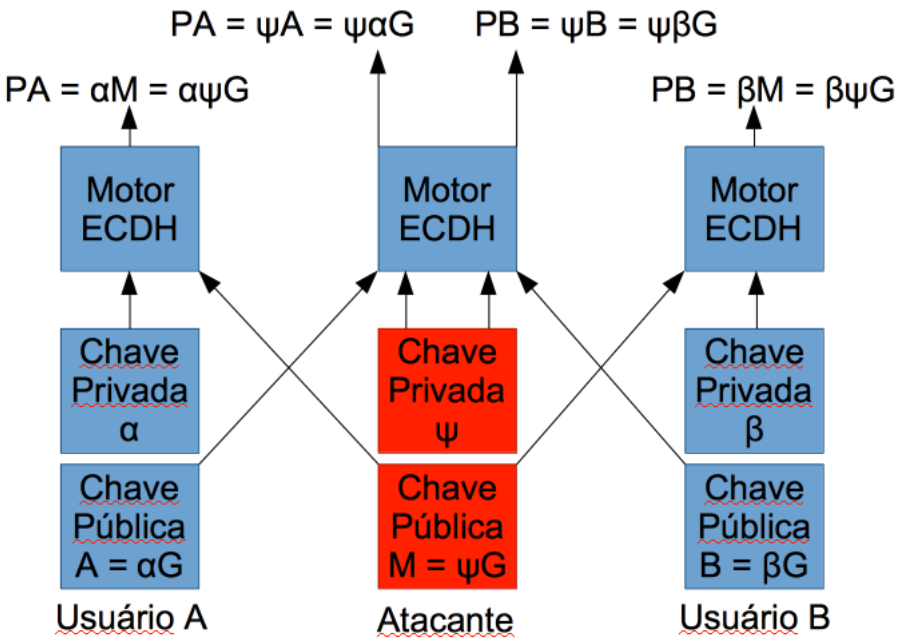


Figura 22 – Ataque de MiM em ECDH

Uma das mais usadas atualmente é a do *Transport Layer Security* (TLS), na qual um cliente que deseja se comunicar com alguma instituição, como um banco por exemplo, solicita um envio de um certificado identificador da instiuição antes da troca de chaves para autenticar a origem da futura troca de chaves. Essa estrutura não mitiga completamente a vulnerabilidade pelo ataque MiM pois um atacante pode rotear a comunicação através de proxy e se passar pela instituição em caso de uma conexão insegura. Outra problema que envolve o monitoramento de dispositivos é que o certificado TLS só garante que a conexão é feita com o servidor emissor do certificado, e dessa forma não autentica os dispositivos que a ele se conectam. No caso dos aplicativos móveis, há uma necessidade de instalação de certificados digitais, que tem prazo de validade, necessitando que o aplicativo seja atualizado constantemente.

O cenário ideal para essa autenticação ocorre quando ambos os lados compartilham alguma informação. Nesse caso o que determina a escolha do algoritmo é a entropia dessa informação compartilhada. Na teoria da informação a entropia é a medida da incerteza associada a uma variável aleatória. Se essa variável aleatória, também conhecida por semente, tiver um tamanho maior que 96 bits, as implementações de Shin et al.(SHIN; KOBARA; IMAI, 2003) ou Saha and Roy Chowdhury(SAHAA; Roy Chowdhurya, 2009) fornecem métodos para autenticação da troca de chaves usuando funções de mão única como as *Secure Hash Algorithm* (SHA).

Entretanto, se o canal pelo qual trafega a mensagem *hash* de autenticação for o mesmo usado para troca de chaves, então o atacante MiM consegue forjar também essa mensagem, e consequentemente realizar a mesma troca de chaves com os usuário do sistema.

Dessa forma é recomendado que na primeira utilização do sistema, um canal adicional para envio de pelo menos uma das mensagens seja empregado. Esse canal é conhecido como canal fora de banda. É muito comum nos aplicativos móveis atuais utilizar um SMS como canal fora de banda, mas no caso de alguns sistemas operacionais isso mantém o risco uma vez que existem APIs nativas da linguagem que provêm recursos para leitura do SMS. Dessa forma um atacante com um malware instalado no smartphone também pode ler esse SMS. Em bancos um canal fora de banda confiável são os caixas eletrônicos levando em consideração que necessitam de uma autenticação do usuário previamente ao envio da mensagem.

Do ponto de vista da usabilidade, para uma primeira utilização de um sistema senhas seriam boas sementes para as funções de mão única de autenticação de troca de chaves. Contudo, atualmente são consideradas seguras para essa finalidade as sementes maiores que 12 bytes. Essa quantidade compromete a usabilidade do sistema pois força o usuário a memorizar uma senha grande. Como muitos sistemas já utilizam senhas menores, o reaproveitamento dessas senhas para um cadastramento único num sistema torna-se uma boa opção considerando a experiência do usuário. Para isso existem os protocolos de troca de chaves autenticados por senha (*Password Authenticated Key Agreement* - PAKE).

Os protocolos PAKE são recentes na literatura da criptografia sendo um dos precursores o *Secure Remote Password* (SRP) (WU, 1998). Atualmente os algoritmos mais o sofisticados não realizam trocas de chaves baseadas em criptografia de chave pública, e permitem que sejam utilizadas senhas de baixa entropia. Em (LANCRENON; ŠKROBOT; TANG, 2016) são discutidos os três principais protocolos do estado da arte do PAKE. Nesse artigo são apresentados dois protocolos mais eficientes que o protocolo JPAKE (HAO; RYAN, 2010), que foi adotado como como padrão no *OpenSSL*. Entretanto, o JPAKE é o único dentre esses que possui uma prova formal de segurança. O J-PAKE troca números inteiros aleatórios, e utiliza o mecanismo de prova de conhecimento zero (*Zero-Knowledge Proof* - ZKP) no qual cada participante sabe que o ZKP foi construído utilizando a senha sem risco de vazamento da senha. No capítulo de protótipos, a implementação do JPAKE é detalhada apresentando as rodadas de troca de mensagem até chegar em um material cripotográfico que pode ser derivado para geração de chave secreta.

2.3 *Web Services REST*

No ano de 2000, Roy Thomas Fielding descreveu em sua dissertação de doutorado (FIELDING, 2000), um novo estilo de arquitetura para sistemas de computação distribuída, o qual nomeou REST. Esse modelo permite que o HTTP seja utilizado de forma eficaz para troca de dados entre softwares remotos. De acordo com FIELDING, se aplicado em conjunto, o REST enfatiza escalabilidade de componentes interativos, distribuição de

componentes independentes, diminuição de latência das interações, segurança reforçada e encapsulamento de sistemas legados.

A Web moderna tem utilizado esse estilo como guia para desenvolvimento de arquitetura de sistemas distribuídos. A abordagem conjunta com padrões conhecidos da Internet como o Hypertext Transfer Protocol (HTTP) e o Uniform Resource Identifiers (URI) consolidou essa arquitetura por ser de grande facilidade de implementação. Os princípios arquiteturais de REST permitem que qualquer tipo de dado seja trafegado pela web, desde um texto simples até dados binários.

2.3.1 Detalhes arquiteturais de *REST*

Endereçabilidade

Cada recurso tem que ser acessível através de um identificador único, ou seja, um único endereço. Dessa forma é possível padronizar a identidade de um objeto, facilitando seu referenciamento. As URIs gerenciam a endereçabilidade em REST. A URI nada mais é do que o endereço que é digitado em um browser quando se quer acessar determinada página. Entretanto em REST, a URI determina um recurso de acordo com o seguinte padrão:

scheme: //host:port/path?queryString-fragment

- scheme – protocolo de comunicação utilizado. Em REST é usado o http ou https;
- host – o nome DNS ou endereço IP;
- port – a porta de comunicação;
- path – uma analogia com os diretórios que consiste no caminho de determinado recurso;
- queryString – atributos que podem ser adicionados a variáveis para determinada programação;
- fragment – utilizado para apontar para determinado local do recurso.

Objetivando acessar uma base de dados do um cliente 1111 de determinada empresa, a criação da URI seria:

https://empresa.intranet.com/cliente/1111

Interface Limitada

Como o REST utiliza o HTTP para trafegar dados, então as operações que podem ser implementadas estão restritas aos seus métodos. Toda programação por trás deve ser

iniciada através de um comando como GET, PUT, DELETE, POST, HEAD ou OPTIONS. Limitar as operações é vantajoso, pois cria um ambiente de programação mais familiar, uma vez que todos os métodos são conhecidos. Outra vantagem é a interoperabilidade, pois se outros sistemas precisarem acessar o web service, é conhecido de antemão quais métodos estão disponíveis. Também podemos destacar a questão do cache da Internet. Se dois programas distintos tentarem fazer uso do mesmo recurso, a segunda solicitação será processada mais rapidamente, aumentando assim o poder de escalabilidade do REST.

Orientação à representação

Esse princípio arquitetural define que em uma operação GET, o recurso será transferido exatamente no estado atual da solicitação. As mudanças de estado são processadas através de métodos PUT ou POST. Dessa forma as alterações dos recursos são totalmente controladas pelo programador.

Comunicação sem estado

Por ser desenvolvido em cima do HTTP, o servidor não grava as sessões de solicitações de dados do cliente, a não ser que o programador implemente tal função. Esse princípio aumenta o poder de escalabilidade do sistema tendo em vista que as operações que gravam estado são computacionalmente caras e aumentaria demais a latência do servidor.

2.3.2 Desenhando os serviços

Atribuindo URIs

É comum utilizar um modelo de objetos, como um diagrama de classes para atribuir URIs aos recursos que serão disponibilizados. As URIs endereçam unicamente determinado recurso, distinguindo-os um dos outros (WEBBER; PARASTATIDIS; ROBINSON, 2010). Para modelar uma carteira de clientes, com endereço e telefones, pode ser utilizado o seguinte padrão:

- Cliente: https://empresa.intranet.com/cliente/id
- Endereço: https://empresa.intranet.com/cliente/id/endereco
- Telefones: https://empresa.intranet.com/cliente/id/telefones/?=*******

Onde id seria o identificador do determinado cliente.

Escolhendo o formato de dados

Determinar o tipo de representação dos dados é muito importante para que o sistema possa se comunicar com outros previamente desenvolvidos. Atualmente os dois

tipos mais populares da Internet são o JavaScript Object Notation (JSON) e o Extreme Markup Language (XML) de acordo com (BURKE, 2010).

Determinando os métodos HTTP

A decisão sobre quais métodos devem ser atribuídos a cada recurso deve ir ao encontro do que se objetiva. Por exemplo, o método GET é por padrão um método somente de leitura e, portanto, deve ser utilizado para tal finalidade exclusivamente. Podemos disponibilizar vários métodos para determinado recurso, porém devemos levar em consideração a real necessidade dessa sobrecarga para evitar confusão em sua utilização. Os métodos mais utilizados são:

- GET: utilizado regularmente para leitura ou apenas visualização do recurso;
- PUT: cria ou atualiza determinado recurso no servidor, sendo um método idempotente, pois não importa quantas vezes é efetuado PUT da mesma solicitação, ele sempre resultará no mesmo objeto;
- POST: cria um novo recurso no servidor gerando um identificador único e devolvendo uma resposta com os dados do novo recurso, por exemplo seu ID;
- DELETE: remove o recurso do servidor;

2.4 Trabalhos relacionados

A abordagem mais recente de autenticação de dois fatores usando NFC em dispositivos móveis foi concebida pela Aliança FIDO. A série de especificações de 2º Fator Universal (U2F) (FIDO Alliance, 2015b) padroniza tokens de autenticação baseados em criptografia de chave pública que pode, entre outros, ser utilizadas através de NFC(FIDO Alliance, 2015a). U2F visa especificamente a autenticação de dois fatores para serviços da web com pares de chaves assimétricas criptografia fortes. O transporte NFC U2F (FIDO Alliance, 2015a) permite que esses pares de chaves sejam armazenados seguraente em tokens físicos dedicados com interface NFC, como exemplo o YubiKey NEO(Yubico,).

Assim como a abordagem U2F, vários cartões de identidade nacional, como o espanhol DNIe (LEóN-COCA et al., 2013) ou o alemão NPA(HORSCH; BRAUN; WIESMAIER, 2011), são utilizáveis para autenticação através de sua interface NFC e, consequentemente, podem ser usados em combinação com smartphones NFC. No entanto, as implementações destes protocolos de autenticação normalmente variam entre diferentes países.

Hölzl et al.(HöLZL et al., 2015) projetaram uma variação do protocolo remoto seguro de senhas (SRP) para autenticação entre as etiquetas e aplicativos seguros NFC. Pode ser possível adaptar abordagens semelhantes para verificação de senha em cartões

inteligentes NFC programáveis. Porém, a senha ainda continua sendo um requisito para todas as autenticações.

No que diz respeito as etiquetas NFC, a empresa Inside Secure oferece sua linha de produtos VaultIC como uma solução contra a falsificação , clonagem e roubo de identidade (Inside Secure, 2015). Com base na criptografia de chave pública de curvas elípticas, essas etiquetas NFC podem ser autenticadas usando um protocolo de desafio-resposta. No entanto, a Inside Secure parece ser restritiva na prestação de informações sobre a forma como esse sistema funciona e as etiquetas não estão facilmente disponíveis.

Urien(URIEN, 2010) descreve uma extensão para o protocolo TLS usando o protocolo Programa de Autenticação de Chip (CAP) da EMV(EMVCo, 1994) para cartões de pagamento , como cartões de crédito e débito para estabelecer uma chave compartilhada para TLS-PSK entre o servidor bancário e navegador do cliente web. Isso pode reduzir a ameaça de ataques man-in-the-middle (MITM) usando certificados TLS forjados ou não confiáveis. De acordo com (URIEN, 2010) este protocolo só pode ser implementado se o servidor tem acesso a um servidor de autenticação emissor, um serviço prestado pelo emissor do cartão, que pode verificar criptogramas EMV-CAP. Por conseguinte, é questionável se esta tecnologia estaria aberta a outras áreas, exceto sites bancários.

O caso de uso de autenticação mais proeminente com tags NFC no contexto da plataforma Android é o Smart Lock. Introduzido no Android 5.0, o Smart Lock desbloqueia a tela de bloqueio , quando na proximidade de outro dispositivo de confiança. Uma das tecnologias que podem ser utilizadas para Smart Lock é NFC. De acordo com Elenkov (ELENKOV, 2014), o mecanismo de desbloqueio geralmente se baseia no número de série livremente legível, por exemplo o identificador anti-colisão de um tag NFC. Consequentemente, ele depende de identificadores simples que poderiam ser facilmente clonados em etiquetas especialmente criadas ou emuladas por emuladores de cartões especiais (DUC et al., 2009). Nenhum mecanismo criptográfico é utilizado para validar a autenticidade das etiquetas. Além disso, os cartões NFC do padrão EMV são de baixo custo.

O objetivo deste estudo foi avaliar as possibilidades de desenhar novos protocolos para a já amplamente usadas etiquetas NFC de baixo custo com o objetivo de estabelecer um canal autenticado e seguro entre o aplicativo móvel e um *Web Service*. As abordagens já existentes se utilizam de tokens NFC de alto custo, por exemplo YubiKey ou NFC Java Cards EMV, ou com baixa disponibilidade, como no caso do VaultIC, ou então com especificações de documentos de identificação de países. Além disso as abordanges atuais centram-se principalmente no uso das etiquetas NFC em mecanismos de autenticação por meio de protocolos de desafio e resposta entre as partes. Em oposição a isso, as arquiteturas apresentadas no próximo capítulo utilizam as etiquetas NFC como um depósito de materias criptográficos usados para estabelecer um canal de comunicação criptografada e autenticada,

Tabela 3 – Resumo dos trabalhos relacionados

Trabalho	Propriedades	Desvantagens
YubiKey NEO	Tokens com criptografia assimétrica	Tokens de custo elevado
DNIe	Identidade nacional espanhola	Utiliza padrões específicos
NPA	Identidade nacional alemã	Utiliza padrões específicos
VaultIC	Etiqueta NFC de desafio-resposta	Baixa disponibilidade
Hoelzl et. al, 2015	Variação de SRP para etiquetas	Não substitui senha
Urien, 2010	Extensão do TLS para chips EMV	Cartões de alto custo
SmartLock	Desbloqueio de tela do Android	Identificadores burláveis

atuando como parte integrante desse canal seguro. Um resumo dos trabalhos relacionados trabrelac é apresentado na Tabela 3.

3 Protocolos Protecting Touch (PT)

O uso da tecnologia NFC para habilitar serviços de computação móvel depende de mecanismos seguros para seu funcionamento. As técnicas apresentadas ao longo do capítulo anterior permitem a criação de soluções funcionais, porém com restrições de disponibilidade ou segurança. Neste capítulo, descreve-se o projeto de dois protocolos de autenticação amigável e segura entre dispositivos móveis e servidores usando NFC. Esses protocolos são denominados Protecting Touch (PT), sendo que o PT1 é aplicado em etiquetas NFC de armazenamento, enquanto o PT2 utiliza etiquetas processadoras do tipo MIFARE DESFire. A Tabela 4 apresenta as definições de símbolos utilizados nos protocolos.

3.1 Protecting Touch 1 (PT1)

Para o protocolo Protecting Touch 1, qualquer etiqueta de armazenamento poderá ser empregada como fator de autenticação. A etiqueta é utilizada para armazenar uma chave compartilhada válida para autenticação entre o aplicativo e seu servidor. O material criptográfico armazenado no smartphone é usado para vincular o conteúdo da etiqueta à uma instância da aplicação instalada. Esse material pode ser tanto simétrico quanto assimétrico para criptografia e autenticação da mensagem NDEF.

Tabela 4 – Símbolos usados nos protocolos

Símbolo	**Definição**
$E_k(m)$	Cifragem de m usando a chave k
$D_k(c)$	Decifragem de c usando a chave k
$HMAC(k, m)$	Codificação de mensagem de autenticação m usando a chave k
$DH_{A,B}(k)$	Troca de chaves autenticada entre A e B usando a chave k baseada no protocolo Diffie-Hellman
$PAKE_{A,B}(p)$	Troca de chaves entre A e B autenticada por senha p
$ROL_n(x)$	Rotação de x para a esquerda em n bits
$SCR(x)$	x definida pelo determinado esquema
$a \mid\mid b$	Concatenação de a com b
r_x	Número aleatório
k_x	Chave compartilhada
$K_{x,Pub}$	Chave pública K pertencente a x
$K_{x,Priv}$	Chave privada K pertencente a x

3.1.1 Leiaute da memória da etiqueta

Os dados para PT1 são empacotados em uma mensagem de NDEF que pode ser armazenada em qualquer tipo de etiqueta NFC. A carga $OTK_{Payload}$ de um registro de uma chave de única utilização (OTK) contém um identificador de usuário (ID_{user}), uma chave de índice de contador (CT_{OTK}), e a chave de uso único (k_{OTK}) . Além disso, a carga útil registro é criptografada e contém um código de autenticação de mensagens. A criptografia e a mensagem de autenticação são feitas usando chaves conhecidas para o aplicativo e armazenadas em (ou protegidas com) um chaveiro (*key vault*) implementado no sistema operacional. Um vetor de inicialização aleatória para encadeamento dos blocos de cifras é mantido na memória do aplicativo.

O processo de formação da carga $OTK_{Payload}$ é representado pelas regras a seguir, sendo que inicialmente se faz a concatenação do identificador, chave de índice e chave de uso único, formando primeiro uma carga limpa, como em:

$$OTK_{plain} = ID_{user} \,||\, CT_{OTK} \,||\, k_{OTK}$$

Essa carga é então cifrada, como em:

$$OTK_{enc} = E_{k_{App,enc}}\left(OTK_{plain}\right)$$

A partir da mensagem cifrada, é gerada uma mensagem de autenticação HMAC:

$$OTK_{mac} = HMAC\left(k_{App,mac}, OTK_{enc}\right)$$

Por fim a mensagem cifrada é concateada com sua mensagem de autenticacão:

$$OTK_{Payload} = OTK_{enc} \,||\, OTK_{mac}$$

Assumindo 4 bytes para ID_{user} e 4 bytes para CT_{OTK}, um comprimento de 32 bytes para k_{OTK}, AES-128 para criptografia, e HMAC-SHA-256 como codificação de mensagem de autenticação (32 bytes), a carga de um registro de OTK ocuparia 80 bytes. Incluindo o cabeçalho NDEF e o nome do registro do tipo "fkaway.br:ptotk" (escolhido), um registro OTK pode caber em 100 bytes. Com HMAC-SHA-512 e AES-256, o registro se encaixa em 130 bytes.

Além do registro OTK , a etiqueta NFC pode conter um registro NDEF para disparar automaticamente o aplicativo móvel. Por exemplo, no Android, isto pode ser conseguido por meio do *Android Application Record* (AAR).

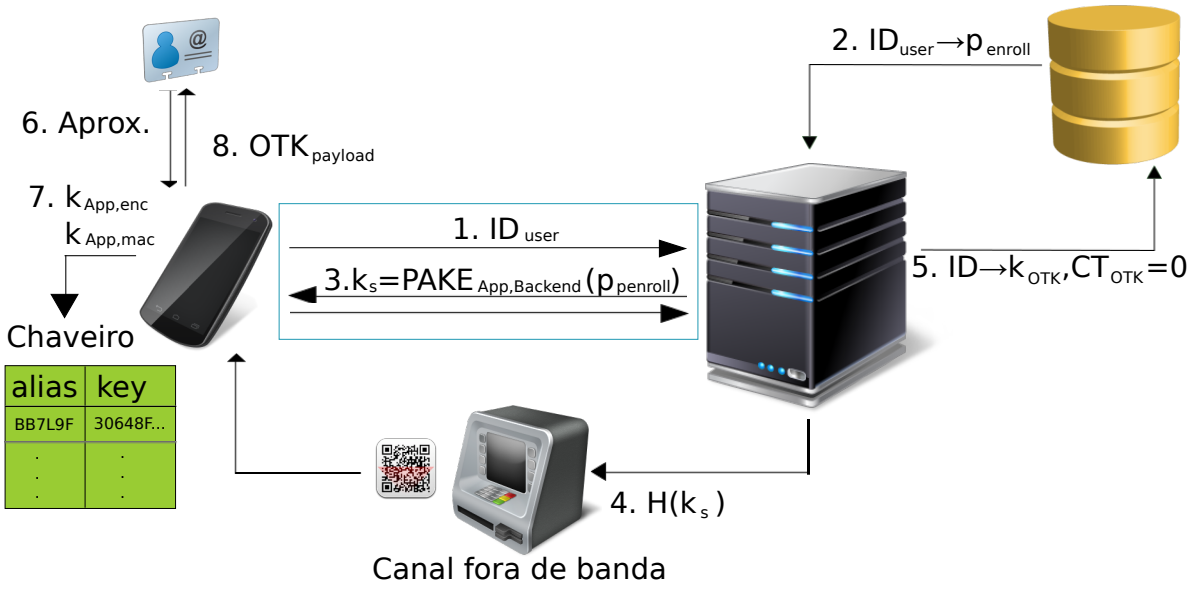


Figura 23 – Cadastramento inicial de etiqueta do Protecting Touch 1

3.1.2 Cadastramento inicial de etiqueta

Durante o cadastramento inicial da etiqueta se armazena nela o identificador de usuário e uma primeira chave de única utilização. Além disso, o aplicativo gera um novo par de chaves e as armazena em seu chaveiro. O protocolo de inicialização é disparado pelo usuário digitando seu identificador de usuário (ID_{user}) e uma senha de cadastramento (p_{enroll}) no aplicativo. O protocolo consiste nas etapas listadas na Figura 23 e descritas a seguir:

1. O aplicativo inicializa a interação com o servidor através do envio do ID_{user}, e indicando o cadastramento de uma nova etiqueta.

   App $\rightarrow$ *Servidor*: ID_{user}

2. O servidor carrega a senha de cadastramento a partir de seu banco de dados (com base no ID_{user}) e começa a fase de autenticação. Se nenhuma correspondência ID_{user} for encontrada, a comunicação com o aplicativo é encerrada.

3. O aplicativo e o servidor realizam uma troca de chaves, autenticada pela senha p_{enroll}.

   Aplicativo $\leftrightarrow$ *Servidor*: $k_S = PAKE_{App,Backend}\left(p_{enroll}\right)$

4. No caso da senha de cadastramento não ser considerada suficientemente forte para algum cenário de aplicação, a fase de autenticação pode ser reforçada através de

um canal fora de banda. Para isso tanto servidor quanto aplicativo geram um hash usando a chave compartilhada k_S. Em seguida, o servidor transmite esse hash para o aplicativo através do canal canal fora de banda. O aplicativo compara os hashes para verificar se ele estava se comunicando diretamente com o servidor. Tal canal fora de banda poderia ser uma máquina de caixa eletrônico, que exibe o hash sob a forma de um código Quick Response (QR) legível pelo aplicativo, e solicitando que o usuário forneça o resultado da verificação, depois de conectado, usando seu cartão do banco e senha.

5. O aplicativo e o servidor criam um contador zerado e derivam k_{OTK} da chave de sessão. O servidor armazena a chave de único uso e o contador inicial em seu banco de dados.

   $$CT_{OTK} = 0$$
   $$k_{OTK} = HMAC\left(k_S, CT_{OTK}\right)$$

6. O usuário aproxima uma etiqueta NFC (depois de ter sido instruído a fazê-lo pelo app) .

7. O aplicativo gera novas chaves simétricas $k_{App,mac}$ e $k_{App,enc}$ em seu chaveiro. Se o chaveiro não suportar chaves simétricas diretamente, um novo par de chaves assimétricas é gerado em seu lugar. Nesse caso chaves aleatórias temporárias, $k_{App,mac}$ e $k_{App,enc}$, são geradas, criptografadas usando a chave pública do par de chaves do chaveiro, e a versão criptografada é armazenado na memória do dispositivo.

8. O aplicativo constrói o novo registro OTK (de ID_{user}, o contador de índice de chave inicial, e a chave de único uso), criptografa e adiciona um código de autenticação de mensagem usando as chaves simétricas do chaveiro ou as chaves simétricas temporárias, como descrito na seção 3.1.1, e o envia para armazenamento na etiqueta.

   Aplicativo $\rightarrow$ *Etiqueta*: $\quad OTK_{Payload}$

3.1.3 Operação do protocolo

O protocolo de autenticação é iniciado aproximando-se a etiqueta NFC do dispositivo móvel. Isso tanto pode acontecer após o aplicativo instruir o usuário a aproximar a etiqueta, quanto o usuário tocar a etiqueta mesmo quando o aplicativo não estiver em execução. O protocolo de autenticação consiste nos seguintes passos apresentados na Figura 24 e descritas a seguir:

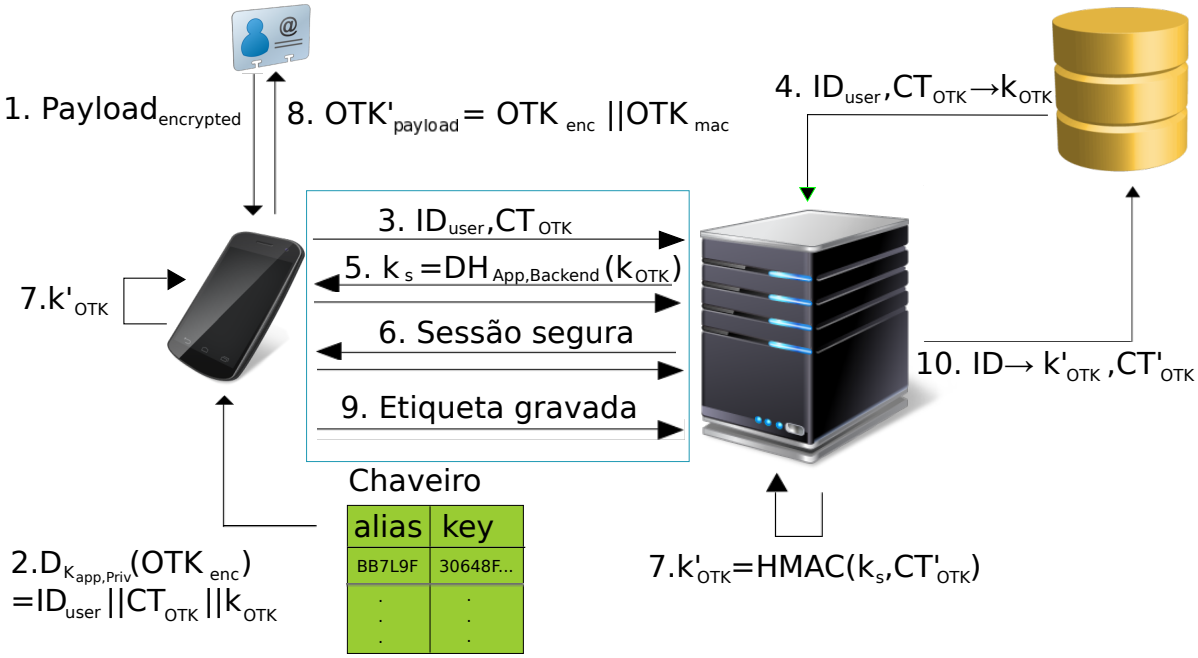


Figura 24 – Operação do Protecting Touch 1

1. O aplicativo lê a mensagem NDEF da etiqueta NFC e extrai o registro OTK.

   Aplicativo $\leftarrow$ *Etiqueta*: $OTK_{Payload}$

2. O aplicativo divide a carga em OTK_{enc} e OTK_{mac}, verifica o código de autenticação de mensagem, decifra o registro de carga OTK, e extrai os elementos de dados (ID_{user}, CT_{OTK}, k_{OTK}) usando as chaves $k_{App,mac}$ e $k_{App,enc}$ armazenadas no chaveiro do sistema operacional.

   $OTK_{Payload} \rightarrow OTK_{enc}$ $OTK_{mac} OTK_{mac} \stackrel{?}{=} HMAC\left(k_{App,mac}, OTK_{enc}\right)$ $OTK_{plain} = D_{k_{App,enc}}(OTK_{enc})$

   $OTK_{plain} \rightarrow ID_{user}$ CT_{OTK} ^{k}OTK

   Se o chaveiro não suportar chaves simétricas diretamente, um par de chaves assimétricas armazenadas dentro do chaveiro pode ser usada. Nesse caso, as chaves simétricas $k_{App,mac}$ e $k_{App,enc}$ são armazenadas criptografadas, na forma $e_{App,...}$, em um espaço inseguro da memória do dispositivo, sem prejuízo da segurança. As chaves só são descriptografadas usando a chave privada $K_{App,Priv}$ do par de chaves dentro do chaveiro, quando necessário.

3. O aplicativo inicializa a interação com o servidor através do envio do ID_{user} e do

CT_{OTK}, a fim de indicar ao servidor qual usuário e etiquetas serão usadas.

Aplicativo $\rightarrow$ *Servidor*: $ID_{user},\ CT_{OTK}$

4. O servidor carrega k_{OTK} do banco de dados (usando ID_{user} e o contador CT_{OTK}) e inicia a fase de autenticação. Se nenhuma chave de único uso for encontrada, a comunicação com o aplicativo é encerrada.

5. O aplicativo e o servidor realizam uma troca de chaves autenticada usando k_{OTK} como chave de autenticação.

   Aplicativo $\leftrightarrow$ *Servidor*: $k_s = DH_{App,Servidor}\left(k_{OTK}\right)$

6. Após estabelecer uma chave de sessão k_s por meio da troca de chaves, o aplicativo e o servidor derivam uma chave usando uma função KDF (Key Derivation Function) (CHEN, 2008) e continuam sua comunicação através de um canal seguro autenticado e criptografado com a chave derivada.

7. O aplicativo e o servidor incrementam o contador e derivam uma nova k'_{OTK} a partir da chave de sessão. A chave é derivada usando um HMAC (HOTP) com base no contador usando o algoritmo apresentado em (M'RAIHI et al., 2005).

   $$CT'_{OTK} = CT_{OTK} + 1$$
   $$k'_{OTK} = HMAC\left(k_s, CT'_{OTK}\right)$$

8. O aplicativo constroi um novo registro OTK (a partir do $ID_{usuário}$, o contador incrementado, e a nova chave de uso único), criptografa e adiciona um código de autenticação de mensagem usando as chaves do chaveiro.

   $$OTK'_{plain} = ID_{user} \,||\, CT'_{OTK} \,||\, k'_{OTK}$$
   $$OTK'_{enc} = E_{k_{App,enc}}\left(OTK'_{plain}\right)$$
   $$OTK'_{mac} = HMAC\left(k_{App,mac}, OTK'_{enc}\right)$$
   $$OTK'_{Payload} = OTK'_{enc} \,||\, OTK'_{mac}$$

   Se o chaveiro chave não suportar chaves simétricas diretamente, novas chaves simétricas $k_{App,mac}$ and $k_{App,enc}$ são geradas e utilizadas para ambas as operações acima. Em seguida, são criptografadas para $e_{App,mac\,/\,enc}$ usando o par de chaves assimétricas do chaveiro e armazenadas na memória do dispositivo.

Em seguida, o registro OTK é armazenado na etiqueta.

$$\text{App} \rightarrow Tag: \quad OTK'_{Payload}$$

9. Após escrever com êxito os dados na etiqueta, o aplicativo envia uma confirmação para o servidor. O usuário é instruído a reaproximar a etiqueta em caso de erros de gravação.

10. O servidor, por sua vez, armazena a nova chave de único uso e o contador de índice incrementado em seu banco de dados e remove a de único uso antiga. Se o aplicativo não confirmar uma operação de gravação bem-sucedida, o servidor descarta a nova chave de único uso e termina a sessão. Depois de um determinado número de falhas pode bloquear o acesso ao sistema, requerendo novo cadastramento.

3.2 Protecting Touch 2 (PT2)

O protocolo Protecting Touch 2 é projetado especificamente para etiquetas MIFARE DESFire. Apesar da versão DESFire EV1 ser a escolha do projeto devido a disponibilidade, o protocolo também opera com outras gerações das etiquetas DESFire após pequenas modificações.

O PT2 usa o protocolo de autenticação de três vias para autenticar mutuamente a etiqueta e o servidor, e para estabelecer uma chave compartilhada entre a etiqueta e o servidor. O protocolo se aproveita de um detalhe de implementação específica do protocolo de comunicação DESFire: o fato de que a resposta do comando de leitura (que é usado para ler um arquivo) é criptografado e autenticado. O aplicativo usa esse fato para obter o conteúdo de dados de um arquivo criptografado pela chave de sessão. Os dados do arquivo criptografado são então usados como uma chave de sessão (vide r_A, r_B, $k_{s,tag}$, e k_s nos itens (5)–(6) da seção 3.2.3) entre o aplicativo e o servidor. A versão de texto plano dos dados de arquivo e a chave de autenticação da etiqueta são considerados segredos de longo prazo compartilhados com servidor.

3.2.1 Leioute da memória da etiqueta

A etiqueta contém uma aplicação DESFire que consiste em dois arquivos. O primeiro arquivo contém o identificador do usuário (ID_{user}) e está configurado para ser livremente legível. O segundo arquivo contém a chave secreta de longo prazo compartilhada (k_{LT}) entre o servidor e a etiqueta e está configurado para ser legível apenas em modo criptografado. Além disso, a aplicação DESFire contém uma chave de autenticação mútua (k_{tag}), no caso com o servidor, usando o algoritmo AES. Autenticação e criptografia

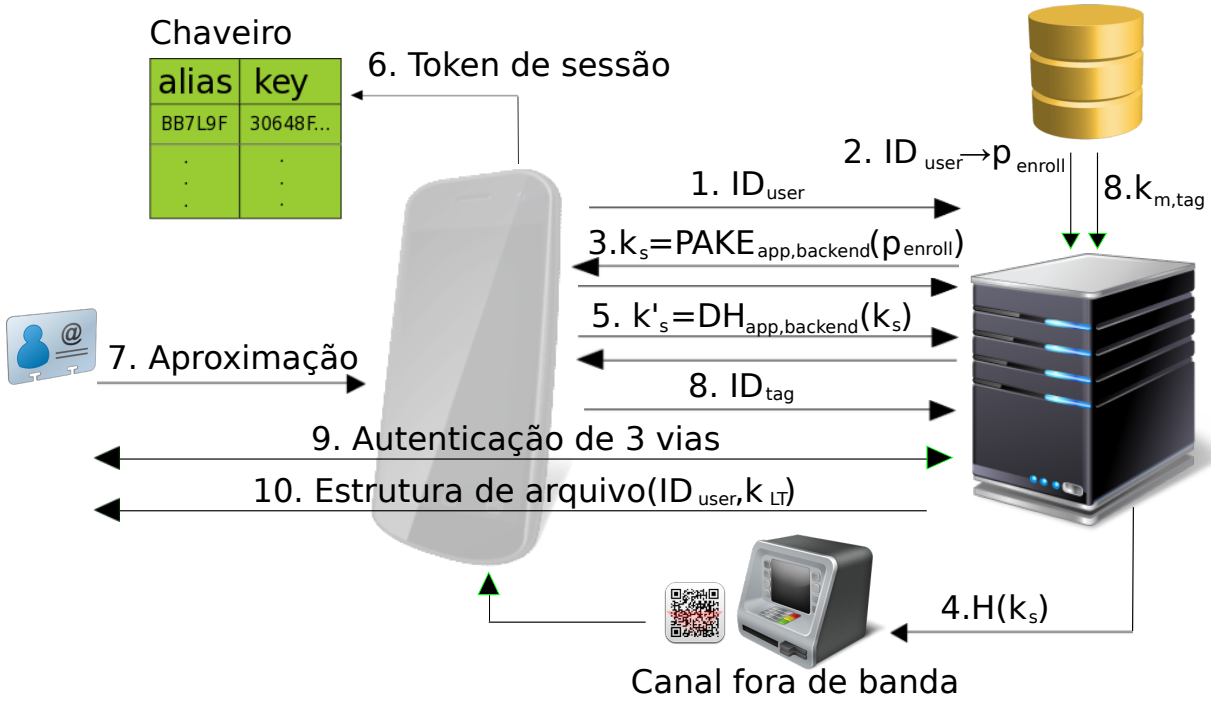


Figura 25 – Cadastramento inicial da etiqueta do Protecting Touch 2

são obrigatórios para ler o segundo arquivo. Além disso, a etiqueta DESFire pode ser configurada como Tipo 4 do Forum NFC, contendo um registro NDEF para o lançamento automático do aplicativo.

3.2.2 Cadastramento inicial de etiqueta

O cadastramento inicial do PT2 cria uma aplicação DESFire e, se necessário, uma mensagem NDEF (vide (NFC Forum, 2011)) em uma etiqueta DESFire. Além disso, o procedimento cria e grava os dois arquivos contendo o identificador do usuário (ID_{user}) e a chave de longo termo compartilhada (k_{LT}), configurando as permissões e a chave de autenticação específica da aplicação DESFire k_{tag}.

Como pré-requisito exige-se que a etiqueta tenha uma chave mestra compartilhada pré-configurada, $k_{m,tag}$, com permissão para criar novas aplicações DESFire. Além disso os usuários devem ter recebido suas credenciais (ID_{user} e senha de cadastramento p_{enroll}) através de algum canal fora de banda, por exemplo correio eletrônico. Abordagens alternativas são discutidas na seção 3.3.

O cadastramento é iniciado pelo usuário digitando seu identificador (ID_{user}) e senha de cadastramento (p_{enroll}) no aplicativo. O protocolo consiste dos seguintes passos ilustrados na Figura 25, que são:

1. O aplicativo inicializa a interação com o servidor enviando ID_{user} e indicando o cadastramento de uma nova etiqueta.

Aplicativo $\rightarrow$ *Servidor*: ID_{user}

2. O servidor carrega a senha de cadastramento do seu banco de dados, com base no ID_{user}, e começa a fase de autenticação. Se nenhuma correspondência ao ID_{user} for encontrada a comunicação com o aplicativo é encerrada.

3. Aplicativo e servidor realizam uma troca de chaves autenticada pela senha p_{enroll}.

   Aplicativo $\leftrightarrow$ *Servidor*: $k_s = PAKE_{App,Backend}\left(p_{enroll}\right)$

4. O mesmo mecanismo do PT1 pode ser usado para executar uma verificação adicional através de um canal fora de banda (ver secção 3.1.2, passo 4).

5. Se a verificação fora de banda for usada, a comunicação é reiniciada com o reenvio do ID_{user} juntamente com uma nova troca de chaves autenticada por k_s, gerando uma nova chave de sessão k'_s. Isso é feito para reduzir os riscos de um possível vazamento de chaves durante o período potencialmente longo de tempo que se leva para concluir a verificação fora de banda.

   Aplicativo $\rightarrow$ *Servidor*: ID_{user}
   Aplicativo $\leftrightarrow$ *Servidor*: $k'_s = DH_{App,Backend}(k_s)$

6. O servidor e o aplicativo trocam um token de sessão, que é utilizado para autenticar a instância de instalação do aplicatico em sessões de comunicação futuras. Embora o mecanismo exato usado para o token de sessão não seja relevante para o protocolo, ele pode ser implementado como um par de chaves assimétricas armazenadas na Android Keystore, por exemplo, com a chave pública armazenada pelo servidor.

7. O usuário aproxima a etiqueta DESFire depois de ter sido instruído a fazê-lo pelo aplicativo.

8. O aplicativo transmite ID_{tag} ao servidor a fim de permitir que a procura pela chave mestra compartilhada da etiqueta $k_{m,tag}$ no banco de dados.

9. Em seguida o aplicativo funciona como um *proxy* transparente entre o servidor e a etiqueta para trocar comandos APDUs necessários para configurar a etiqueta e seu conteúdo de memória, por meio da autenticação mútua de três vias utilizando a chave mestre $k_{m,tag}$.

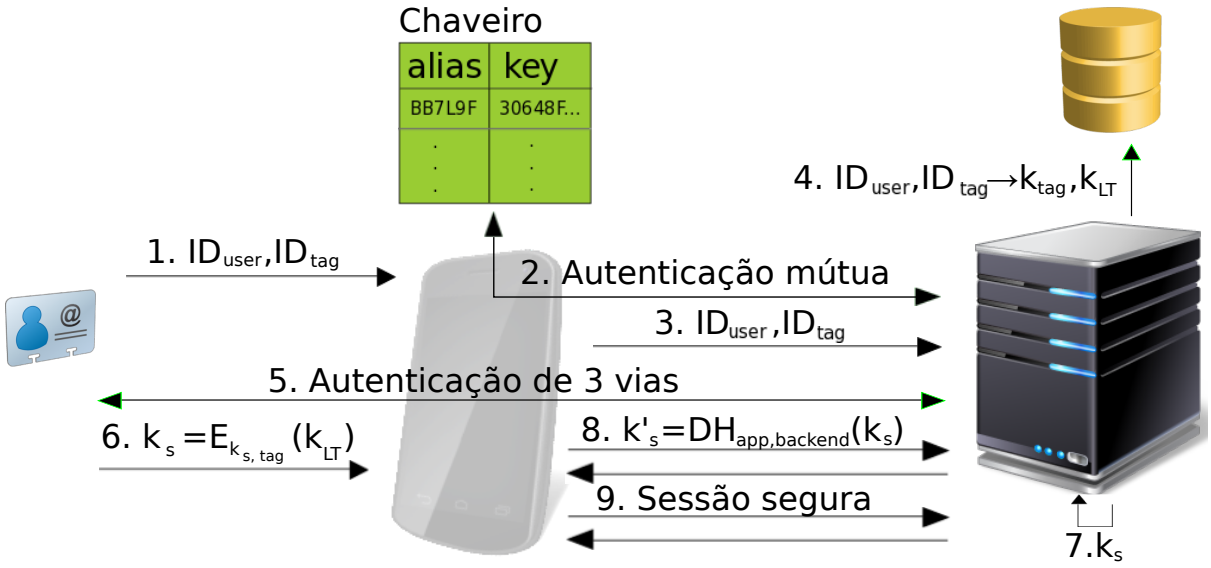


Figura 26 – Operação do protocolo Protecting Touch 2

10. Por fim, a configuração das aplicações DESFire, incluindo a chave k_{tag}, a estrutura do arquivo, os dados do arquivo contendo ID_{user}, o arquivo cifrado contendo k_{LT}, e todas as permissões de acesso, é armazenada na etiqueta.

3.2.3 Operação do protocolo

O protocolo de autenticação é iniciado aproximando-se a etiqueta NFC ao dispositivo móvel. Isso tanto pode acontecer após o aplicativo instruir o usuário a aproximar a etiqueta, como quando o usuário tocar a etiqueta, mesmo se o aplicativo não estiver em execução. O protocolo de autenticação segue os passos apresentados na Figura 26, descritos a seguir:

1. O aplicativo lê ID_{user} e ID_{tag} (identficador anti-colisão) da etiqueta.

   Aplicativo $\leftarrow$ *Etiqueta*: ID_{user}, ID_{tag}

2. O aplicativo e o servidor se autenticam mutuamente usando um token de sessão previamente estabelecido, armazenado no chaveiro acessível pelo aplicativo. Embora o mecanismo exato usado para o token de sessão não seja relevante para o protocolo, no protótipo apresentado no próximo capítulo ele foi implementado como um par de chaves assimétricas na Android Keystore. O aplicativo usa a chave privada para autenticar sessões e o servidor armazena a chave pública deste par, durante o cadastramento, para a verificação da autenticação de sessão.

3. O aplicativo inicia a fase de autenticação com o servidor enviando ID_{user} e o ID_{tag}, a fim de indicar ao servidor qual usuário e etiqueta serão usados para se autenticar.

Observe que ID_{user} também é necessário para correlacionar a sessão.

Aplicativo $\rightarrow$ *Servidor*: $ID_{user},\ ID_{tag}$

4. O aplicativo obtém k_{tag} e k_{LT} do banco de dados baseados no ID_{user} e ID_{tag}, e inicia a fase de autenticação com a etiqueta. Se nenhuma chave correspondente é encontrada no banco de dados, a comunicação com o aplicativo é encerrada.

5. O servidor inicia a autenticação AES de três vias com a etiqueta, enviando comandos APDUs apropriados para o aplicativo. O aplicativo funciona como um *proxy* transparente entre o servidor e a etiqueta.

   Servidor $\rightarrow$ *Etiqueta*: $Iniciaautenticação$
   Servidor $\leftarrow$ *Etiqueta*: $E_{k_{tag}}(r_B)$
   Servidor $\rightarrow$ *Etiqueta*: $E_{k_{tag}}(r_A \mid\mid ROL_8(r_B))$
   Servidor $\leftarrow$ *Etiqueta*: $E_{k_{tag}}(ROL_8(r_A))$

   Primeiro, a etiqueta gera um valor aleatório r_B, criptografa usando a chave k_{tag}, e retorna o criptograma para o servidor. O servidor decifra o valor recebido, gera uma sequência hexadecimal aleatória r_A, concatena com a versão rotacionada do valor recebido r_B, e envia esse valor concatenado criptografada com a chave compartilhada k_{tag}. A etiqueta, por sua vez, decifra o valor recebido e verifica r_B para autenticar o servidor. Após a verificação bem sucedida, a etiqueta retorna uma versão rotacionada e criptografada de r_A. Finalmente, o servidor decifra o valor recebido e verifica r_A para autenticar a etiqueta.

   No final da autenticação mútua, etiqueta e servidor chegam em uma chave de sessão efêmera r_A and r_B construida de acordo com o protocolo DESFire.

6. O aplicativo lê o arquivo que contém a chave secreta de longo prazo (k_{LT}) e obtém sua forma criptografada k_s a partir da chave de sessão $k_{s,tag}$ compartilhada entre a etiqueta e o servidor.

   Aplicativo $\leftarrow$ *Etiqueta*: $k_s = E_{k_{s,tag}}\left(k_{LT}\right)$

7. O servidor também calcula k_s usando $k_{s,tag}$ e k_{LT}. Nesse ponto ambos conhecem a chave de sessão k_s.

8. Aplicativo e servidor realizam uma troca de chaves autenticada por k_s a fim de impedir que intrusos espionando a comunicação NFC obtenham uma chave com poder para decifrar futuras comunicações.

Aplicativo $\leftrightarrow$ *Servidor*: $k'_s = DH_{App,Backend}(k_s)$

9. Após estabelecer uma chave de sessão k'_s, o aplicativo e o servidor continuam sua comunicação através de um canal seguro (autenticado e criptografado por meio da chave de sessão estabelecida).

3.3 Estratégias alternativas de cadastramento

Para o nosso cenário atual de cadastramento, assumimos que a etiqueta está pré-configurada com uma chave mestra compartilhada $k_{m,tag}$. Isto ocorre, por exemplo, quando as tags DESFire já foram distribuídas para os usuários em algum outro caso de uso de aplicação, como um cartão de transporte público, um token de controle de acesso, ou um cartão empresarial, dentre outros Nesses casos, é necessária a cooperação com o operador do sistema existente, a fim de obter acesso à chave mestra ou delegar a configuração inicial de etiqueta (passo 9 na seção 3.2.2) àquele operador. As etiquetas poderiam até ser completamente pré-cadastradas antes da distribuição e serem entregues aos usuários em estado pronto para uso. Isso eliminaria a necessidade do cadastramento inicial pelo aplicativo.

A última geração de etiquetas MIFARE DESFire, DESFire EV2, possuem gerenciamento de aplicações, o que pode simplificar significativamente o cadastramento de etiquetas já utilizados para outras aplicações e emitidos por outros prestadores de serviços.

3.4 Implementação de protótipo

Para executar os protocolos conforme a especificação do capítulo anterior, os protótipos seguem o fluxo especificado na Figura 27, e descrito a seguir:

1. A aproximação da etiqueta inícia a operação dos protocolos, podendo ser na tela inicial do Android ou na *Main Activity* (tela principal do aplicativo):

   - No caso de a etiqueta ser aproximada na tela inicial, o sistema operacional Android recebe os dados da etiqueta encapsulados e os distribui para as aplicações que registraram em seus *Manifests* para tratar as mensagens NFC. Esse mecanismo é denominado *Android Dispatch System* (ADS). Se um AAR compuser uma mensagem NDEF encapsulada na etiqueta, então o ADS se encarregará de inicializar o aplicativo e o entregará toda a mensagem NDEF.
   - Quando a etiqueta é aproximada com o aplicativo em execução, uma nova *Intent* é disparada contendo a mensagem NFC encapsulada. Para que a *Activity* que

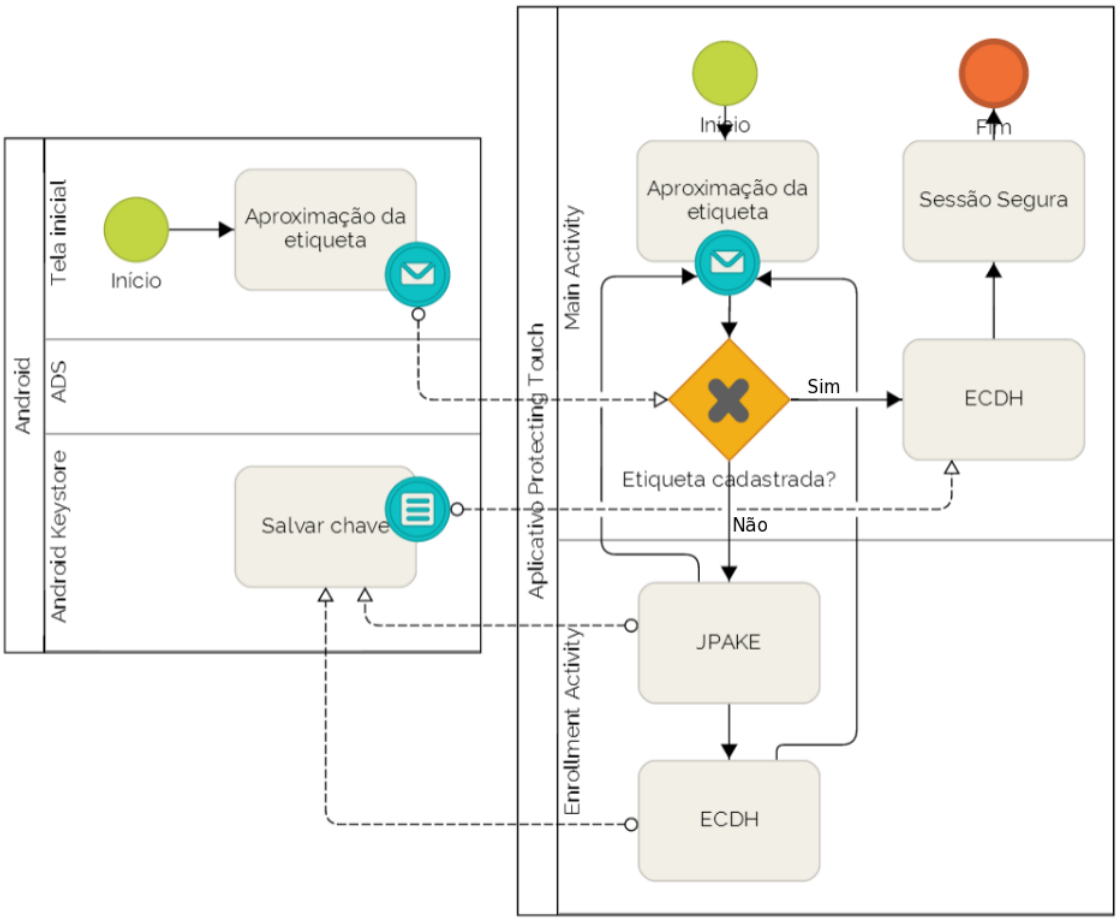


Figura 27 – Fluxograma das Activities do Protótipo do PT

estiver presente na tela do dispositivo possa receber essa mensagem, o Android provê o recurso de *Foreground Dispatch*. Esse recurso deve ser habilitado por toda *Activity* de forma a impedir que o ADS distribua a mensagem NDEF para outra aplicação, ou que o AAR faça a aplicação ser reinicializada.

2. Se a etiqueta não estiver cadastrada, a *Enrollment Activity* é iniciada.

3. A fase de cadastramento começa com a execução do protocolo JPAKE representado na Figura 29. No protocolo JPAKE implementado no BC, cada participante deve manter uma instância da classe *JPAKEParticipant* durante as três rodadas de troca de informações até chegarem na chave final autenticada e confirmada. Esse objeto não é serializável, ou seja, não é possível salvar uma instância em um arquivo ou no banco de dados. Isso se torna um problema de implementação em Web Services REST, tendo em vista que um dos seus conceitos principais é o de não armazenar estado durante as requisições. Contudo, em JAVA é possível contornar esse problema usando a anotação *@Singleton*, que faz com que uma única instância da classe dos três serviços, que tratam as rodadas, seja executada do servidor. Numa aplicação real, o servidor estará atendendo a diversas requisições simultanteamente, e portanto se

faz necessária uma estrutura para armazenar os dados de diversos participantes, que no caso do protótipo é a *HashMap*. Essa estrutura armazena os vários participantes atrelando-os a um índice. O índice para os protocolos PT é o identificador do usuário. Após completadas as três rodadas do JPAKE, os dados do participante são descartados da estrutura, e consequentemente da memória. Numa situação real, dificilmente muitos participantes estariam executando o cadastramento dos PTs ao mesmo tempo, já que o cadastramento representa o primeiro uso. Sendo assim, essa solução atende a escalabilidade do sistema de forma razoável mesmo contando somente com um servidor.

Nos protótipos o JPAKE foi implementado até a segunda rodada. A partir desse passo a chave de sessão *SK* pode ser derivada do *keyMaterial* da instância do *JPAKEParticipant*. Dependendo do caso de uso em alguma aplicação real, pode ser necessária implementação da terceira etapa. Contudo, como dito anteriormente, a confirmação é implícita. A terceira rodada tem objetivo único de confirmar se ambos os lados possuem a mesma chave. Se ambos os lados não possuírem a mesma chave, então será impossível decifrar uma futura troca de mensagens.

4. No PT1 só existe um passo de troca de chaves, e portanto, o próximo passo é salvar a chave gerada usando a Android Keystore. No PT2 após a conclusão da primeira troca de chaves, segue o ECDH autenticado pela chave resultante da primeira troca de chaves, salvando a nova chave na Android Keystore. A Android Keystore passa a suportar armazenamento e operações com chaves simétricas a partir da SDK versão 23 correspondente ao Android 6.0. De acordo com os dados do Android Studio, somente 4,7% dos celulares no mundo estão equipados com essa versão do Android, e portanto foi descartado o uso desse nível de SDK. Contudo, a partir da SDK 18, a Android Keystore opera com chaves assimétricas RSA. Essa exclusividade da versão de SDK, quanto ao tipo de chave, impacta diretamente no protoloco PT1. Como não há armazenamento seguro de chaves simétricas, um par de chaves é gerado na Android Keystore no passo 7 do cadastramento. A chave pública criptografa $k_{App,mac}$ e $k_{App,enc}$, e a cifra resultante é armazenada em um arquivo inseguro. Uma vez que somente a chave privada armazenada na Keystore tem o poder para descriptografar, não há problema de clonagem desse arquivo inseguro em outro dispositivo. Dessa forma, o passo 2 da operação do protocolo faz uso da chave privada para decifrar o conteúdo do arquivo e obter $k_{App,mac}$ e $k_{App,enc}$.

   Para as trocas de chaves com ECDH, a BC dispõe das curvas elípticas consideradas seguras e padronizadas pelo NIST. No projeto foi utilizada a curva *brainpoolp256r1* que permite acordar entre as partes uma chave AES de até 256 bits. Com o intuito de evitar ataques de MiM, todas as trocas de chaves são autenticadas. Como a entropia da semente é alta, mensagems hash SHA-256 das chaves públicas PK_a (aplicativo) e

PK_S (servidor) são trocadas entre o aplicativo e o servidor na forma da Figura 28. As sementes são k_{OTK} no PT1, e k_S no PT2. A última mensagem hash que trafega do servidor para o aplicativo, de forma a confirmar a autenticidade da chave do servidor, é a que deve ser enviada num canal fora de banda no passo 4 do cadastramento do PT2.

Outra diferença entre os dois protocolos do ponto de vista da tecnologia NFC é o formato de troca de mensagens. A área segura do DESFire, utilizada no PT2, é acessível somente por meio de comandos APDUs. O Android possui a classe nativa *IsoDep* que permite o envio desses comandos. Na implementação do protótipo do PT2, as cargas de dados da autenticação de três vias e a estrutura de arquivos a serem gravadas na etiqueta são recebidas através dos serviços REST, e então o comando é construído pelo aplicativo mobile. Para construção de protótipo do PT1 não é necessário envio de comandos APDUs para o cartão. Como descrito na seção 3.1.1, os dados são empacotados em mensagens NDEF. A partir do nível de API 9 da SDK do Android, são incluídas as classes *NfcAdapter*, que faz a comunicação com o chip NFC, e *NdefMessage*, que é a abstração da mensagem NDEF contendo registros da classe *NdefRecord*. No protótipo do PT1 somente dois registros compõem a mensagem NDEF: o $OTK_{Payload}$ cifrado pela chave da Android Keystore e o AAR disparador da aplicação.

5. Com a etiqueta cadastrada o aplicativo é direcionado novamente para a tela principal. A partir desse passo, o protocolo de operação se inicia com um novo ECDH agora autenticado com a chave da Android Keystore.

6. Por fim, a sessão segura é estabelecida. A partir desse passo os aplicativos que implementarem essa rotina podem fazer uso dessa sessão segura.

3.5 Análise de segurança

A segurança dos protocolos PT reside na combinação de múltiplos fatores de autenticação para a criação de chaves de sessão, como, a etiqueta NFC, o aplicativo móvel com um chaveiro, bem como nos protocolos de comunicação com o servidor. Esses fatores podem ser individualmente atacados na tentativa de tomar o controle de um dos lados do canal de comunicação. Note que a segurança do sistema servidor não é escopo desse projeto, e já existem diversos estudos e aplicações que garantem sua segurança. Assim, nas próximas páginas serão avaliados os possíveis ataques ao processo de comunicação.

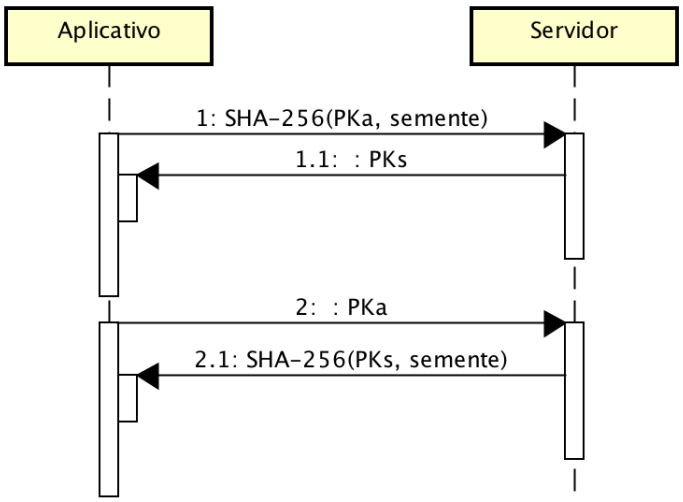


Figura 28 – Diagrama de sequência da troca de chaves autenticada por semente

3.5.1 Ataques à etiquetas NFC

Um atacante pode se apoderar do conteúdo da etiqueta NFC clonando toda sua memória. Esse é um risco potencial para o PT1 uma vez que o tipo de etiqueta a ser empregada pode não prover nenhuma proteção contra clonagem. Entretanto, os dados armazenados estão criptografados com uma chave acessível somente por uma instância específica instalada no smartphone. Clonagem ou até mesmo o roubo da etiqueta por si só não representa risco para os usuários do sistema. Em outras palavras, o conteúdo da etiqueta é atrelado a um smartphone específico, e portanto, é inútil para o atacante que não possua acesso ao telefone. Isso reduz, por exemplo, o risco quando comparado ao esquema de autenticação baseada em mensagens SMS uma vez que nos protocolos propostos o atacante precisa manter o controle de duas peças separadas de hardware. No entanto, é reconhecido que os usuários tendem a carregar ambos os componentes ao mesmo tempo, o que resulta em ser possível que ambos os componentes sejam roubados juntos.

Um cenário para um potencial ataque na arquitetura do PT2 envolve a comunicação entre a etiqueta e o aplicativo. Entretanto, a comunicação é criptografada por chaves compartilhadas entre a etiqueta e o servidor, fazendo com que a aplicação não tenha a capacidade de decifrar as mensagens. Esse é o comportamento esperado já que a chave de longo prazo do cartão nunca deve ser armazenada no smartphone. A mensagem criptografada é usada como base para estabelecer um canal seguro entre o aplicativo e o servidor (ver passo 6 na seção 3.2.3). Um intruso em proximidade da comunicação NFC pode interceptar essa chave. Entretanto, a menos que ele também intercepte ativamente o protocolo de troca de chaves entre o cliente o servidor ao mesmo tempo, essa informação se torna infrutífera para o atacante.

As primeiras versões do DESFire eram vulneráveis, principalmente pelo uso de

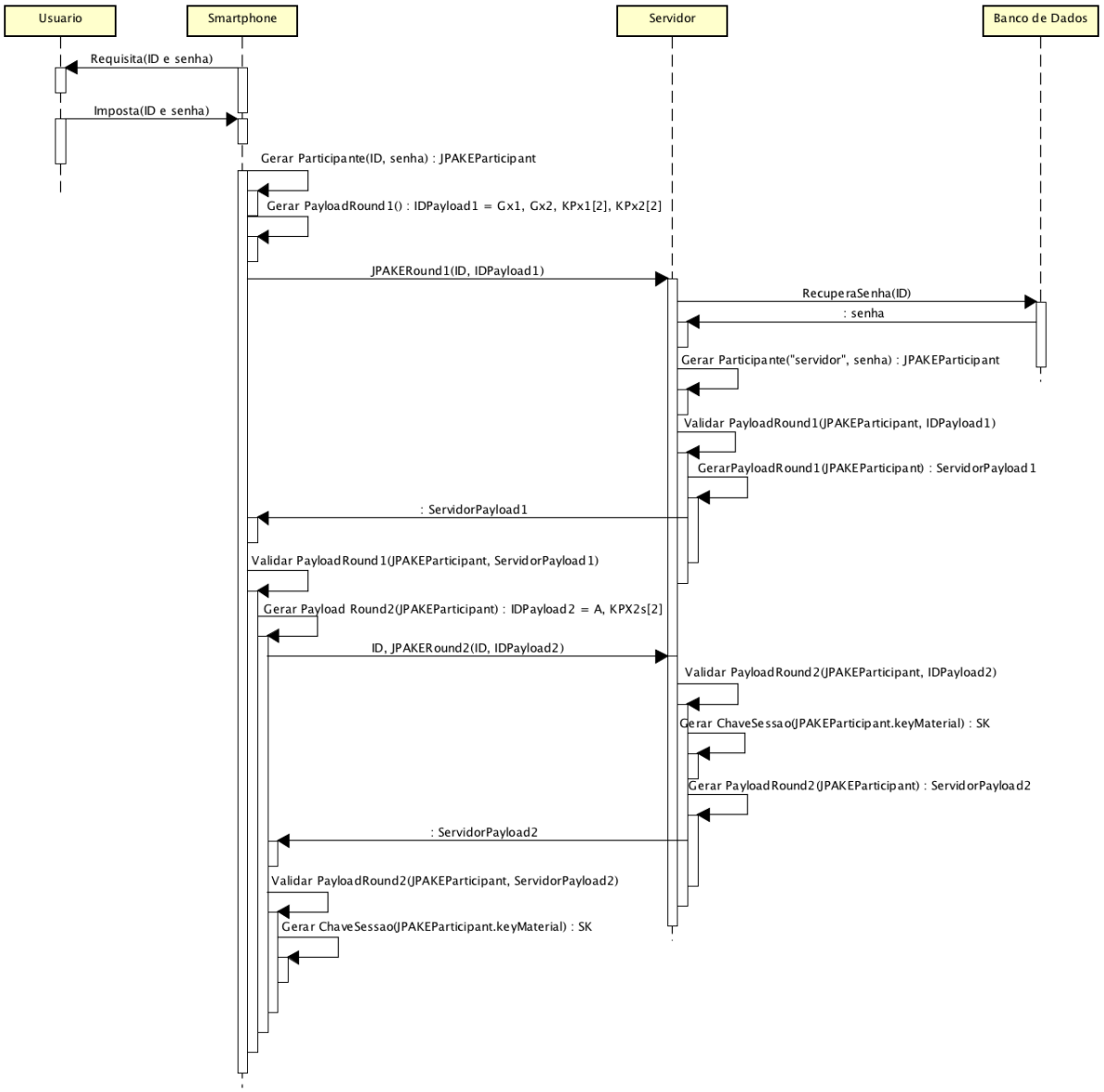


Figura 29 – Diagrama de sequência do JPAKE

sistemas criptográficos com ataques amplamente conhecidos, como o DES. Em (OSWALD; PAAR, 2011) esses ataques, aplicados especialmente ao DESFire, são discutidos. A versão empregada no protocolo é no mínimo a DESFire EV1, cujos ataques potenciais apresentados por Oswald and Paar foram mitigados.

3.5.2 Ataques à aplicação cliente visando o dispositivo

Um atacante com alvo no dispositivo móvel somente pode tentar atacar o chaveiro usado para armazenar os tokens de sessão que criptografa ou descriptografa as informações da etiqueta no PT1. A segurança dessas chaves depende da implementação do chaveiro. Um chaveiro implementado em hardware naturalmente provê um nível maior de segurança do que um implementado em software, já que as chaves ficam armazenadas em um hardware

seguro como em um ambiente de execução seguro (Trusted Zone) ou em um elemento seguro. Entretanto, mesmo se os atacantes conseguirem invadir e controlar o telefone, e tiverem acesso acesso as chaves no chaveiro, eles também teriam que acessar a etiqueta NFC para fazer uso do material criptográfico. Isso implica que aplicativo nunca deve armazenar chaves secretas de longo prazo da etiqueta para manter as futuras comunicações em sigilo no caso do dispositivo for comprometido ou roubado.

Um aplicativo malicioso pode tentar interceptar a comunicação com a etiqueta NFC. Por exemplo, em um dispositivo Android, um aplicativo com acesso ao NFC pode ler qualquer etiqueta. Para o PT1, isso permitiria ao adversário clonar todo o conteúdo da etiqueta, ou realizar um ataque de negação de serviço ao apagar os dados da etiqueta. No PT2, seria impossível que o aplicativo malicioso obtivesse qualquer dado criptográfico da etiqueta.

Comparado com outros métodos, como senhas ou autenticação por SMS, um acesso único à etiqueta é de uso limitado ao atacante. Uma senha, se coletada através de um aplicativo malicioso, permanece utilizável até que o usuário a substitua. Um aplicativo registrado para receber mensagens SMS pode se apoderar do mecanismo de autenticação enviado ao dispositivo. Através dos protocolos propostos, a etiqueta somente pode ser utilizada por meio explícito de interação do usuário (quando um usuário expliciamente aproxima a etiqueta no campo de leitura do seu dispositivo). Esse é o único caso em que o usuário deseja ser autenticado. Consequentemente, o uso da etiqueta NFC melhora a segurança em relação ao uso somente de senha ou mensagem SMS devido a necessidade de interação explícita do usuário ("human-in-the-loop").

3.5.3 Ataques ao canal de comunicação entre o aplicativo e o servidor

Com base nos estudos teóricos assume-se que as primitivas criptográficas empregadas para troca de chaves, autenticação e cifragem de troca de dados são seguras. Autenticar o canal entre o aplicativo e o servidor contra ataques de MiM é um dos principais focos dos protocolos Protecting Touch.

Em ambos os protocolos o material criptográfico usado para autenticação é armazenado somente na etiqueta NFC. No PT1 esse material só pode ser decifrado pelo telefone do usuário. Semelhantemente, no PT2, somente o telefone que está em posse do token de sessão pode acessar o material criptográfico da etiqueta. Dessa forma, o atacante necessitaria ter controle de ambos os dispositivos (smartphone e etiqueta) para poder se apoderar do canal de comunicação entre o aplicativo e o servidor.

Se o ataque de MiM ocorrer durante o cadastramento do PT1 ou PT2, o atacante precisaria conhecer a senha de cadastramento p_{enroll}. Usar um protocolo apropriado de troca de chaves autenticado por senha como o JPAKE, previne potenciais ataques de força

bruta durante a troca de chaves. Adicionalmente, para ajudar a proteger contra os ataques de MiM, pode ser usado o canal fora de banda para enviar um hash da chave secreta k_s estabelecida após a comunicação inicial. Nesse caso o usuário terá uma autenticação adicional para verificação (como a inserção do cartão bancário e senha no caso de uso em aplicativos móveis bancários) no canal fora de banda. Sendo assim, para um ataque bem sucedido no cadastramento, o adversário teria que controlar também o canal fora de banda.

Os protocolos PT1 e PT2 propostos também provêem segurança futura para a comunicação entre o aplicativo e o servidor. Se a chave de sessão k_s vazar no PT1, o adversário poderia decifrar a sessão na qual a chave está sendo usada. Porém, como a chave de sessão é derivada de uma troca de chaves usando a cifra de uso único k_{OTK} como chave de autenticação compartilhada, não é possível gerar a próxima chave de sessão k'_s apenas conhecendo a chave atual. Nem mesmo usando uma chave k_s seria possível descriptografar sessões previamente gravadas utilizando o protocolo DH com chaves efêmeras. A geração de novas chaves a cada sessão faz com que a segurança futura seja proporcionada. A chave de uso único k_{OTK} é lida da etiqueta e usada para proteger ataques MiM. Um adversário com acesso à chave de sessão seria capaz de gerar k'_{OTK} apenas para a próxima sessão, mas para comprometer a comunicação da próxima sessão, ele teria ainda que conseguir interferir no canal de comunicação durante o processo de troca de chaves.

No PT2 a semente para a chave de sessão para comunicação entre o aplicativo e o servidor é gerada diretamente na etiqueta, derivada da chave secreta de longo prazo k_{LT}. Essa derivação é feita cifrando k_{LT} com uma chave de sessão efêmera $k_{s,tag}$, estabelecida entre a etiqueta e o servidor. Isso efetivamente gera uma chave de sessão randômica por k_{LT} e pela chave efêmera $k_{s,tag}$. Levando em consideração que $k_{s,tag}$ é randômica e gerada independentemente, isso também implica que cada nova sessão é independente das chaves de sessões anteriores. Dessa forma, a segurança futura é proporcionada pelo PT2. A segurança futura seria comprometida se a chave de autenticação de longo prazo da etiqueta k_{tag}, junto com a chave secreta de longo prazo k_{LT} fossem comprometidas. Assim, para manter a segurança, o ideal é que essas chaves (k_{tag} e k_{LT}) nunca trafeguem em texto plano.

3.6 Comparação entre o PT1 e o PT2

Tanto PT1 como PT2 foram desenvolvidos no espectro das etiquetas NFC de baixo custo. Ambos protocolos tem diferentes *trade-off* entre disponibilidade, usabilidade e segurança. Enquanto o PT1 pode ser implementado com qualquer etiqueta NFC que possua a quantidade mínima de memória, o PT2 requer a etiqueta específica Mifare DESFire. Mesmo assim, a DESFire tem boa disponibilidade a baixo custo. Etiquetas NFC

adequadas ao PT1 estão disponíveis a aproximadamente 2 reais, enquanto as etiquetas para o PT2 partem de aproximadamente 10 reais com base nos preços encontrados no site Tagstand ((Tagstand, 2016)) e nfc-tag.de ((nfc-tag.de, 2016)). Em comparação, um token dedicado para autenticação NFC YubiKey NEO é vendido a aproximadamente 160 reais.

As etiquetas usadas no PT1 podem ser facilmente clonadas por um atacante com acesso físico ou com a capacidade de inteceptar a comunicação NFC, enquanto a DESFire oferece uma boa proteção contra clonagem, por meio de sua memória segura e com leitura cifrada e autenticada. Apesar de a etiqueta clonada no PT1 ter seu uso limitado (conforme section 3.5.1), tem-se que o PT1 é menos seguro que o PT2. Além disso, como não há proteção contra gravação irreversível nas etiquetas NFC, PT1 é suscetível a ataques de negação de serviço por atacantes maliciosos ou se acidentalmente sobrescrever a memória da etiqueta por outro aplicativo por exemplo.

No aspecto da experiência do usuário, PT1 tem a vantagem da etiqueta ter que ser lida somente uma vez para iniciar o ciclo do protocolo, e escrita somente uma vez no final do ciclo do protocolo. Outra vantagem é que o PT1 permite que as interações do usuário sejam desacopladas da latência de rede, o que não é permitido no PT2, uma vez que a cada aproximação da etiqueta se faz necessária uma nova autenticação.

Outra vantagem do PT1 é que o HMAC dos dados do OTK permite que o aplicativo autentique offline as etiquetas antes de se conectar ao servidor. Isso pode ser também usado para implementar a funcionalidade de desbloqueio do aplicativo, que é menos crítica e não precisa de comunicação com o servidor. No PT2, o aplicativo atua como um proxy para as etapas da autenticação entre a DESFire e o servidor. Consequentemente, a etiqueta necessita estar continuamente próxima ao smartphone até que toda troca de mensagens seja completada.

A atual implementação do PT1 e PT2 considera somente um aplicativo por etiqueta. Isso significa que os usuários precisam carregar uma etiqueta para cada aplicativo. Entretanto, os protocolos podem ser facilmente remodelados para se adaptar a cenários de múltiplos aplicativos por etiqueta. No PT2, cada aplicativo pode ter seu própria aplicação DESFire na etiqueta.

4 Avaliação dos Protocolos

Os protocolos descritos no capítulo anterior foram implementados através de dois protótipos para o sistema Android. Detalhes da implementação e da avaliação desses protótipos são encontradas neste capítulo, bem como as justificativas para a escolha do ambiente Android.

4.1 Definição de Ambiente

Os protótipos para os protocolos Protecting Touch foram desenvolvidos para o ambiente Android por diversas razões. Entre os motivos dessa escolha é possível destacar:

- Dos três sistemas operacionais mais utilizados para smartphones, Android, iOS e Windows Phone, tem-se que o Android ocupa uma fatia de mercado superior a 80%, de acordo com um levantamento feito da utilização do aplicativo do Banco do Brasil.
- O Android possuem uma API para NFC aberta, permitindo fácil implementação dos comandos de controle das etiquetas NFC.
- O Android, em suas versões mais recentes, também apresenta uma API nativa para chaveiro.
- O Android é utilizado por uma ampla gama de fabricantes de dispositivos móveis.
- O iOS (segunda maior parcela de mercado) não tem API disponível para que terceiros criem aplicativos usando a tecnologia NFC.
- O Windows Phone tinha, até recentemente, apenas um modelo de smartphone usando a tecnologia NFC.

Assim, a opção natural para avaliar os protocolos aqui propostos foi pelo sistema operacional Android. Apesar de suas vantagens, o Android também apresenta alguns problemas, que acabaram por restringir as versões em que o protocolo funciona. Sabe-se que versões de API anteriores (até a 18) apresentavam problemas de vulnerabilidade, corrigidas na versão 19. Adotar a versão 19, como feito, restringe o uso dos protocolos PT a cerca de 40% dos smartphones usando Android, o que é uma parcela razoável considerando-se que o uso dos protocolos demanda que se evite riscos já conhecidos.

Além da definição da plataforma móvel a ser utilizada é preciso definir alguns aspectos ligados ao servidor que atenderá aos aplicativos móveis. As decisões tomadas,

Tabela 5 – Ferramentas utilizadas na prototipação

Ferramenta	**Nome**	**Versão**
IDE Android	Android Studio	2.1.1
IDE Servidor	Eclipse EE	4.5
Linguagem de programação	Java	1.8
API Android	Android SDK	19
Web Service REST	Jersey	1.10
API criptográfica	Bouncy Castle	1.52
Banco de Dados	MySQL Server	5.6

tanto para servidor quanto aplicativo, aparecem na Tabela 5, sendo que dela se destacam o uso de Java e da API Bouncy Castle para criptografia.

A escolha pela API Bouncy Castle para criptografia se baseou na sua maior abrangência. Nela podem ser encontradas funções para o protocolo Diffie-Hellman com curvas elípticas (ECDH), hashes SHA 256 e 512, cifras AES e o protocolo JPAKE para troca de chaves com senhas de baixa entropia. Além disso, existe uma versão da API para o sistema Android (Spongy Castle), o que viabiliza a compatibilidade entre os protocolos utilizados em cada ambiente.

4.2 Testes

Os protótipos foram testados em três modelos diferentes de smartphone. Na tabela 6, apresentam-se as configurações relevantes dos modelos de smartphones. A escolha dos smartphones levou em consideração primeiramente se haviam antenas NFC compatíveis com os dois tipos de etiquetas usadas: Mifare Classic e DESFire. Alguns smartphones não são compatíveis com as etiquetas Mifare Classic, notadamente aqueles que possuem os Chipsets da Broadcom, e portanto foram descartados. O segundo fator de escolha foi o processador para averiguar se as operações criptográficas (computacionalmente dispendiosas) poderiam ser processadas em tempo viável. O último fator foi a versão do sistema operacional sendo uma diferente para cada smartphone para verificar a compatibilidade dos protótipos. O tipo de implementação da Android Keystore também é apresentado na tabela 6, bem como seu mês de lançamento no mercado mundial. Os smartphones mais recentes tem sua implementação da Android Keystore em hardware, ou seja, suas chaves criptográficas ficam armazenadas em espaços de hardware reservados, diferentemente da implementação em software, que armazena as chaves em sistema de arquivos.

Para simular o servidor foi utilizado um computador Apple Macbook Pro com um processador Intel Core i7 quad core de 2,2GHz com 6MB de cache L3, e memória RAM de 16 GB 1600 MHz DDR3. A versão do sistema operacional do computador é a OS X 10.11.4 (El Capitan). O banco de dados MySQL estava instalado na própria máquina.

Tabela 6 – Configurações dos modelos de smartphones empregados nos testes

	Sony Xperia Z3	LG G4	Samsung Galaxy S3
Processador	Quad-core 2.5GHz	1.8 GHz 6 Core	1.4 GHz Quad Core
Memória RAM	3 GB	3GB	1GB
Versão do Android	5.1 (API 22)	6.0 (API 23)	4.4 (API 20)
Chipset NFC	NXP PN547	NXP PN547	NXP PN544
Android Keystore	Hardware	Hardware	Software
Lançamento	Setembro/2014	Abril/2015	Maio/2012

Tabela 7 – Tempos de execução dos protocolos (média ±desvio-padrão em ms)

		Sony Xperia Z3	LG G4	Samsung Galaxy S3
PT1	Cadastramento	803,7 ±189, 26	1040,1 ±95, 14	1.941,55 ±235, 63
	Operação	618,65 ±71, 29	492,15 ±84, 64	1.656,75 ±127, 11
PT2	Cadastramento	1352 ±264, 48	1671,45 ±246, 75	2582,2 ±327, 12
	Operação	628 ±233, 77	417,45 ±44, 05	793,3 ±125, 02

Todos os testes foram feitos em rede local.

O procedimento de medição de desempenho consistiu na coleta de 20 amostras de tempo para cada smartphone, realizando tanto a atividade de cadastramento como a de operação da aplicação. Esse número de amostras foi suficiente para que os valores apresentados a seguir fossem estáveis.

4.2.1 Resultados

O desempenho dos protocolos, do ponto de vista de tempo de execução, foi medido para os três dispositivos apresentados na Tabela 6. Levando em consideração um hardware mais novo, espera-se que o melhor desempenho seja do LG G4, seguido pelo Sony Xperia e pelo Galaxy S3.

Os resultados globais de desempenho são vistos na Tabela 7, sendo que os tempos mostrados são em milissegundos. Nela é possível observar que a expectativa de desempenho foi correspondida, pois enquanto o Galaxy S3 leva aproximadamente 2s para o cadastramento (no protocolo PT1), os demais aparelhos consumiram 0,8s (Sony) e 1,0s (LG) para o mesmo protocolo. Deve ser observado aqui embora o Sony tenha melhor desempenho no cadastramento, seu desempenho é pior na operação do aplicativo.

Para identificar os motivos da diferença de comportamento dos aparelhos da Sony e LG foi preciso detalhar como o tempo para execução do protocolo se dividia entre suas principais tarefas. Para tanto as tarefas de cada protocolo foram encapsuladas em threads, medindo-se o tempo gasto para cada thread. Os resultados dessas medições aparecem na Tabela 8. A tarefa que levou mais tempo foi a execução do protocolo JPAKE. Isso já era esperado considerando que a primeira carga do JPAKE tem 3.770 bytes e a segunda carga

Tabela 8 – Tempos de execução das principais tarefas (média ±desvio-padrão em ms)

	Sony Xperia Z3	LG G4	Samsung Galaxy S3
JPAKE	401,45 ±50, 66	765 ±73, 81	1106 ±157, 60
ECDH	206,55 ±51, 37	355,65 ±125, 44	640,45 ±121, 63
Gravar NDEF	173,6 ±5, 69	157,2 ±35, 6	181 ±91, 8
Ler NDEF	0 ±0	0,35 ±1, 13	0,9 ±1, 37
Decifrar OTK	73,8 ±3, 72	21,75 ±5, 4	102,4 ±51, 20
Construir OTK	5,3 ±3, 34	4,8 ±1, 50	7,05 ±2, 01
Aut. DESFire (3 vias)	160,9 ±77, 47	145,55 ±70, 89	375 ±305, 21
Criar Arq. DESFire	16,85 ±1, 95	48,75 ±7, 61	32,55 ±1, 50
Gravar Arq. DESFire	114,55 ±5, 81	172,2 ±36, 69	214,95 ±12, 69

é de 1.890 bytes. O tempo médio mais baixo para essa tarefa de aproximadamente 0,4 s e o mais alto de 1,1 s foram para o Z3 e S3, respectivamente. A segunda tarefa mais custosa, o ECDH autenticado por semente, durou em média aproximadamente metade do tempo em relação ao JPAKE para os aparelhos testados. Um tempo menor era esperado uma vez que a quantidade de dados trafegada no ECDH é bem menor que no JPAKE. O *hash* chave pública contém somente 32 bytes, e a chave 184 bytes. Nessas operações, o Z3 foi mais eficiente, o que justifica o tempo total menor para os métodos de cadastramento dos protocolos. Nas operações criptográficas da Android Keystore, o G4 teve melhor desempenho. Nesse caso o que influenciou não foi o poder de processamento e sim o tipo de implementação. Como o G4 é o único aparelho embarcando o Android 6.0, ele é o único a usar chaves simétricas para criptografia, que são conhecidamente mais eficientes, tanto no processamento quanto no consumo de memória, que as operações de chaves assimétricas das versões anteriores. Isso somado a uma maior quantidade de núcleos fez ele completar todas as tarefas das operações dos protocolos mais rapidamente que os outros celulares.

Outro aspecto avaliado nos protótipos foi o consumo de recursos pelo sistema. Para atender essa demanda três quesitos foram mensurados: necessidade de banda de rede, uso do processador e memória. A necessidade de banda máxima foi o principal foco nessa fase de testes, pois um uso demasiado da rede aumentaria significativamente o tempo total das operações do protocolo no caso de uso em redes de internet móveis de baixa velocidade. O uso do processador é outro item relevante pois se o aplicativo ocupar muito o processador, pode acarretar numa necessidade de uso dedicado, e consequentemente diminuiria a usabilidade num ambiente no qual muitos aplicativos estão executando simultaneamente. Por último, o gasto de memória também pode influenciar negativamente na adoção da solução se for excessivo. As figuras 30, 31 e 32 apresentam os gráficos de uso de memória, CPU e rede, durante execuções sequenciais de cadastramento e operação dos protocolos para os três smartphones testados. Os gráficos foram extraídos do aplicativo Android Studio.

Os picos de necessidade máxima de banda foram registrados durante a execução

do JPAKE, cujo tráfego de dados é maior. Portanto, é no cadastramento dos protocolos PT que a taxa mais alta de consumo de banda foi verificada. No protocolo PT1, a maior taxa de upload registrada foi de 26,82 kilobytes por segundo (KB/s), e de download foi 23,82 KB/s. Já para o PT2, a maior taxa de upload foi 24,04 KB/s, e de download foi 25.20 KB/s.

Para o consumo de CPU, o Android Studio fornece um percentual do seu uso. A medida da CPU registrou seus picos de uso quando precisou liberar memória que o sistema operacional aloca para o aplicativo. Nesse caso, após o acúmulo de algumas transações, o uso da CPU chegou a atingir aproximadamente 45% da sua capacidade máxima. Mas em geral, tanto para o cadastramento quanto para as operações, a CPU é ocupada em menos de 20% de seu potencial, mesmo no processador inferior do Galaxy S3.

A quantidade de memória teve uma variação muito grande entre os smartphones. Enquanto o LG G4 precisou de quase 60 MB de memória para executar cada protótipo, o Xperia Z3 não precisou nem de 40 MB, e o Galaxy S3 não passou dos 20 MB. Isso mostra que as novas versões dos sistemas operacionais demandam mais memória para executar as mesmas tarefas.

4.3 Análise dos Resultados

A implementação dos protocolos usa métodos já existentes para a parte de criptografia, computacionalmente mais robusta, Isso diminui os riscos de implementacão indevida de determinadas rotinas de cifragem e embaralhamento. Os ambientes de programação disponíveis para as escolhas das ferramentas feitas são colaborativos.

Os testes foram executados em versões do Android que atingem 76,9% do total de smartphones equipados com Android de acordo com a tabela 9. Foram desconsideradas as versões com *exploits* conhecidos, ou seja, descartadas as com API 18 ou abaixo. A tendência é de aumento no nível de aparelhos com o sistema operacional suficiente, já que novos lançamentos devem vir equipados com as mais recentes.

O tempo de execução máximo, tanto para a parte de cadastramento quanto para a de operação dos protocolos PT, faz com que o uso deles seja mais rápido que inserir uma senha ou então procurar um aplicativo e inicializá-lo. O que pode aumentar consideravelmente esse tempo é velocidade da conexão com a internet. Nas conexões de internet móvel, a velocidade de upload e download são diferentes. A Tabela 10 apresenta as velocidades de conexão mais comuns no Brasil. Levando em consideração a maior taxa de upload registrada de 26,82 KB/s (equivalente à 214,56 Kbps ou 0,21456 Mbps) a conexão 2G (EDGE) adicionaria um atraso considerável devido ao gargalo da banda. Em 3G ou 4G a largura da banda é suficiente para evitar atrasos significativos tanto para o download ou upload de todos os protocolos.

Tabela 9 – Distribuição cumulativa de uso das APIs pelo Android Studio

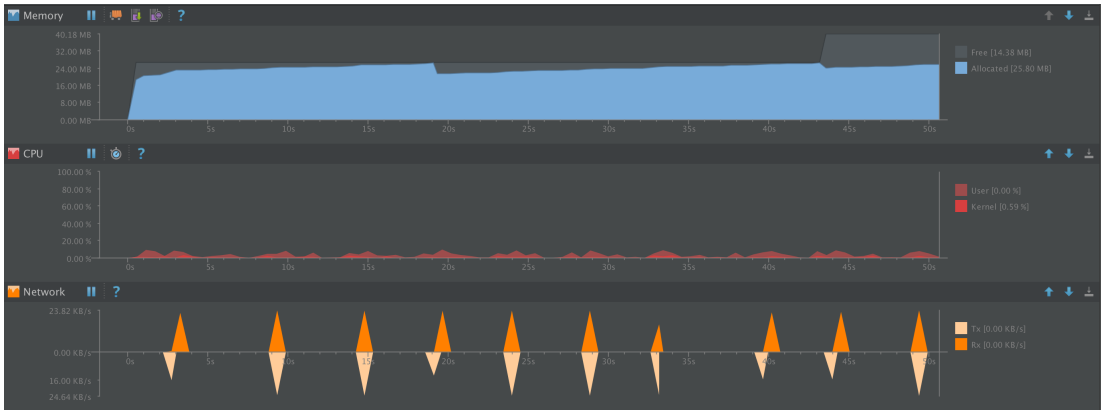
Versão	Nome	Nível de API	Distribuição cumulativa
2.3	Gingerbread	10	100,0%
4.0	Ice Cream Sandwich	15	97,4%
4.1	Jelly Bean	16	95,2%
4.2	Jelly Bean	17	87,4%
4.3	Jelly Bean	18	76,9%
4.4	KitKat	19	73,9%
5.0	Lollipop	21	40,5%
5.1	Lollipop	22	24,1%
6.0	Marshmallow	23	4,7%

Tabela 10 – Velocidades de conexão de internet móvel

	Download	Upload
2G (EDGE)	236,8 Kbps	59,2 Kbps
3G (HSPA)	14,4 Mbps	5,76 Mbps
4G (LTE)	100 Mbps	50 Mbps

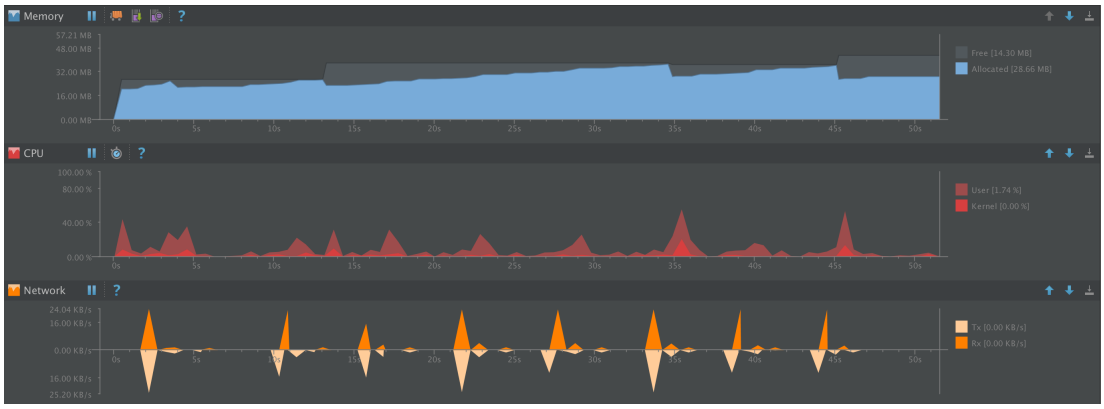
Os testes da necessidade de processamento mostram que não é necessária uma capacidade processamento alta para os padrões atuais dos smartphones. Mesmo em um smartphone considerado mais antigo, como o Galaxy S3, conseguiu-se executar as tarefas eficientemente. Os métodos dos protocolo por si só não consumiram nem 20% do potencial dos processadores testados.

Os novos smartphones geralmente possuem mais de 1 GB de memória RAM. Mesmo no caso de consumo de memória, o total alocado para o programa nunca passou de 60 MB. Sendo assim, em todos os aspectos testados, os protótipos tiveram êxito na execução das tarefas.

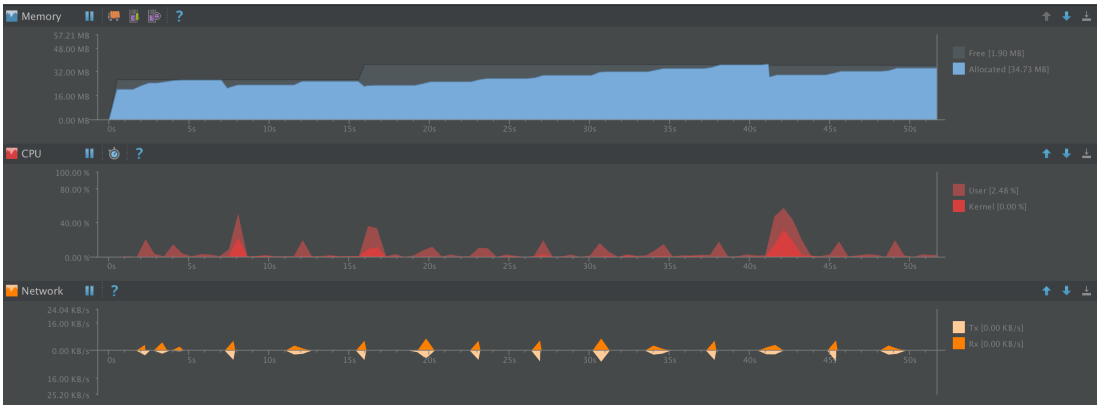


(a) Cadastramento do PT1

(b) Operação do PT1



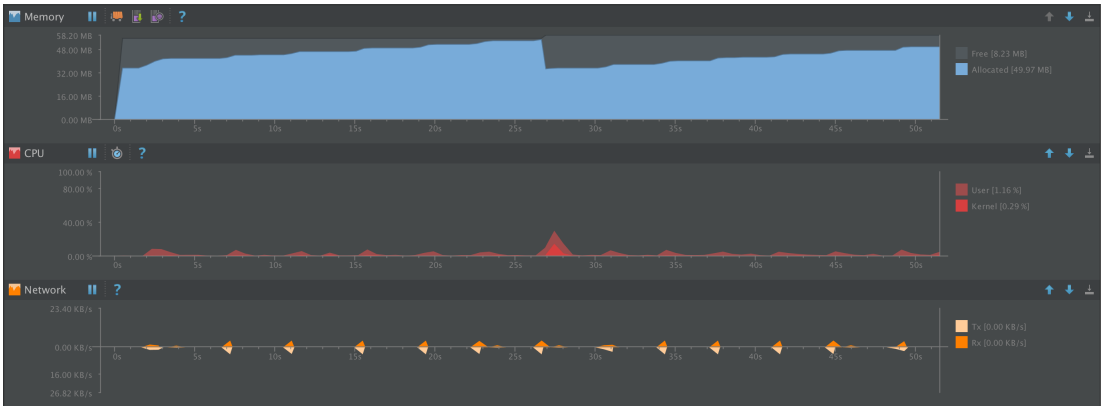
(c) Cadastramento do PT2



(d) Operação do PT2

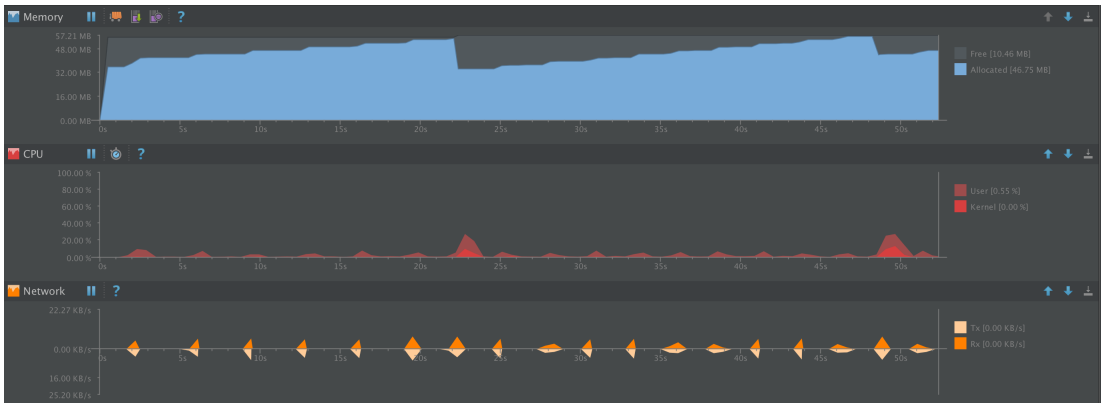
Figura 30 – Grafícos de consumo de recursos do Xperia Z3

(a) Cadastramento do PT1



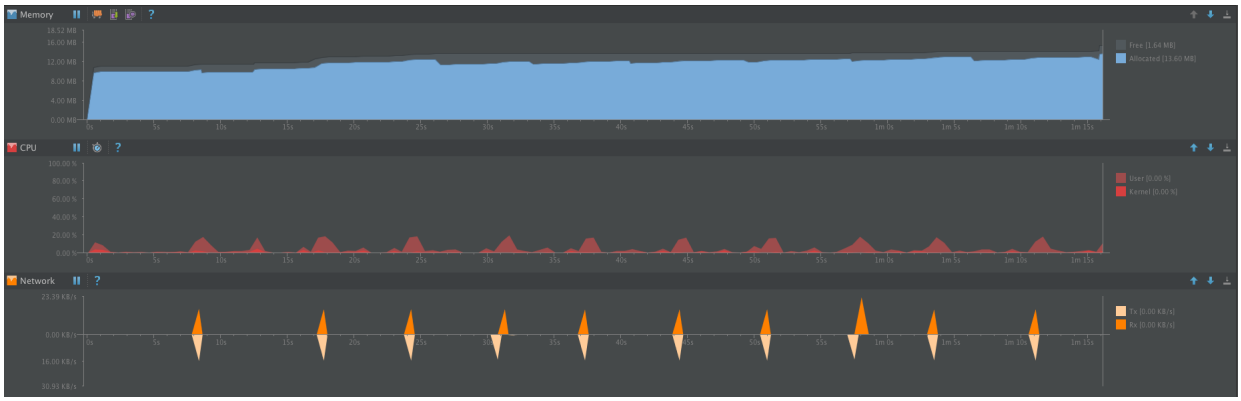
(b) Operação do PT1

(c) Cadastramento do PT2



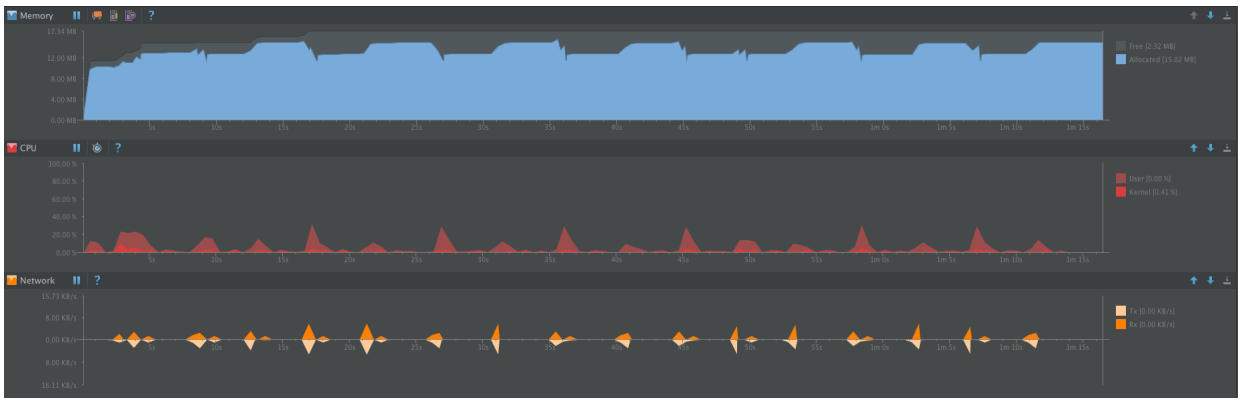
(d) Operação do PT2

Figura 31 – Grafícos de consumo de recursos do LG G4

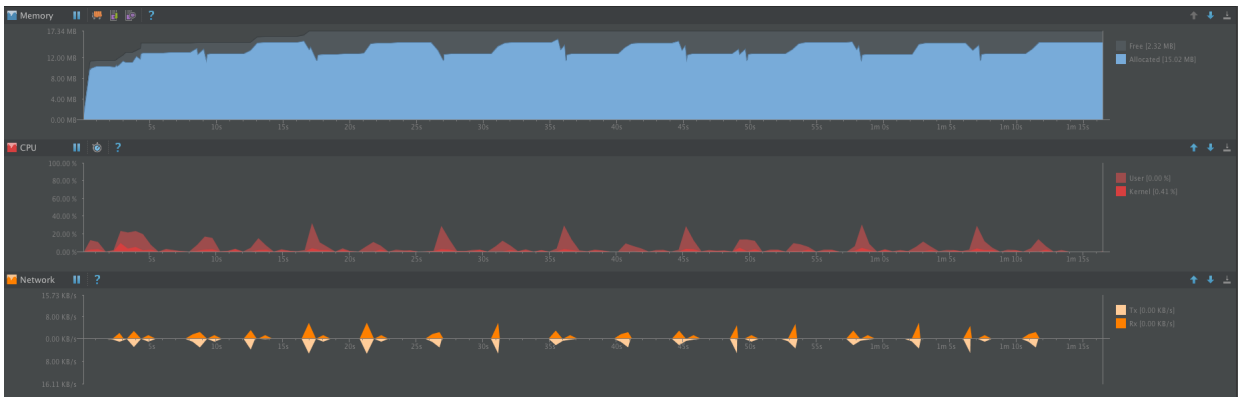


(a) Cadastramento do PT1

(b) Operação do PT1



(c) Cadastramento do PT2



(d) Operação do PT2

Figura 32 – Grafícos de consumo de recursos do Galaxy S3

5 Conclusão e direções futuras

Neste trabalho foi avaliado o uso de etiquetas NFC, de baixo custo e alta disponibilidade, como tokens de autenticação. Para essas etiquetas foram criados dois modelos conceituais de autenticação para comunicação entre dispositivos e servidores, os protocolos Protecting Touch. Em oposição aos mecanismos já existentes, a abordagem desse trabalho usa material criptográfico extraído das etiquetas NFC para elevar a segurança de toda a comunicação entre o smartphone e os serviços Web.

5.1 Análise final

Os dois protocolos foram analisados quanto as suas propriedades de segurança para verificar seus riscos de implementação. Como usualmente os smartphones requerem outro recurso de autenticação para desbloqueio de tela (incluindo forçar a exigência do mecanismo), em muitos casos os protocolos PT podem ser empregados como único mecanismo de segurança.

Do ponto de vista de usabilidade, os testes de velocidade indicam que as operações dos protocolos demandam um tempo que pode ser considerado desprezível. O único ponto que pode pesar contra a solução é o uso de redes de internet de baixa qualidade, tanto em latência quanto em banda de tráfego. Em redes de alta qualidade, os protocolos oferecem uma experiência prática de aproximar a etiqueta NFC ao dispositivo e o aplicativo ser aberto quase que instantaneamente.

O consumo de recursos do dispositivo móvel também foi mensurado. Observou-se que mesmo em aparelhos com hardware desatualizado (2012), o smartphone usa pouco de sua capacidade total para execução dos protocolos. O desempenho é afetado pelas escolhas dos algoritmos para cada passo dos protocolos, porém, existe flexibilidade para sua substituição para atingir níveis mais baixos de consumo.

Dessa forma, é possível concluir que os protocolos fornecem uma boa alternativa ao uso de senhas, provendo segurança e melhor usabilidade, uma vez que é mais rápido aproximar o cartão do que digitar um login, e lembrar e digitar uma longa (forte) senha.

5.2 Direções futuras

Como trabalho futuro, os protocolos podem ser disponibilizados como bibliotecas abertas para uso público. Isso contribuiria de forma significativa para os desenvolvedores de aplicativos móveis, que frequentemente cometem erros ao implementar rotinas criptográficas

em seus aplicativos, como indicado em uma análise de mais de onze mil aplicativos feita em (EGELE et al., 2013).

Ainda no escopo deste trabalho, um terceiro protocolo pode ser desenvolvido, o PT3, usando etiquetas programáveis, como o Java Card. Apesar do PT2 poder ser adaptado a esse tipo de cartão, é possível modelar um protocolo que se adapte ainda melhor a sua estrutura, uma vez que *applets* podem ser programadas especificamente. Esse tipo de cartão já é comum em bancos estrangeiros, e atendem o padrão EMV.

Há campos a explorar na tecnologia NFC. É comum ver etiquetas NFC para controle de acesso em estacionamentos. Nesse caso a catraca eletrônica emite um cartão quando o usuário entra no estacionamento. Quando ele deseja sair, deve comparecer a um dos guichês dentro do shopping para validar o cartão em uma máquina com tecnologia NFC, pagando a tarifa referente ao período que o veículo ficou estacionado. Após a validação é possível clonar o cartão e sair do estacionamento com quantos veículos quiser até o horário limite pago pelo usuário. Atualmente esse procedimento é bem simples pois vários programas de clonagem de etiquetas NFC estão disponíveis gratuitamente pelas lojas virtuais de aplicativos. Essa brecha de segurança se estende a todas as etiquetas NFC que não implementam sistemas criptográficos dinâmicos. Dessa forma, um estudo para mitigar essa vulnerabilidade pode também ser foco de um futuro trabalho.

Referências

ABDUL, D. S.; ELMINAAM, H. M. A. K.; HADHOUD, M. M. Performance evaluation of symmetric encryption algorithms. *Communications of the IBIMA*, v. 8, 2009. Citado na página 31.

BARCLAYS. *Contactless Mobile*. 2012. <http://www.barclays.co.uk/contactless-mobile>. Accessed: 2015-09-30. Citado na página 14.

BURKE, B. *RESTful Java with JAX-RS*. [S.l.]: O'Reilly Media, 2010. Citado na página 43.

CHEN, L. Recommendation for key derivation using pseudorandom functions. *NIST special publication*, v. 800, p. 108, 2008. Citado na página 51.

CHOU, W. Elliptic curve cryptography and its applications to mobile devices. 2003. Citado na página 35.

COSKUN, V.; OK, K.; OZDENIZCI, B. *Professional Application Development for Android*. [S.l.]: Wrox, 2013. Citado na página 17.

COSKUN, V.; OK, K.; OZDENIZCI, B. *Professional NFC application development for Android*. [S.l.]: John Wiley & Sons, 2013. Citado na página 14.

DAEMEN, J.; RIJMEN, V. Aes proposal: Rijndael. Citeseer, 1998. Citado na página 31.

DUC, D. N. et al. Open issues in RFID security. In: *Proceedings of the 2009 International Conference for Internet Technology and Secured Transactions (ICITST 2009)*. [S.l.]: IEEE, 2009. p. 1–5. <http://dx.doi.org/10.1109/ICITST.2009.5402510>. Citado 2 vezes nas páginas 20 e 44.

EGELE, M. et al. An Empirical Study of Cryptographic Misuse in Android Applications. In: *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security (CCS '13)*. [S.l.]: ACM, 2013. p. 73–84. <http://dx.doi.org/10.1145/2508859.2516693>. Citado na página 76.

ELENKOV, N. Dissecting Lollipop's Smart Lock. *Android Explorations Blog*, dez. 2014. <http://nelenkov.blogspot.com/2014/12/dissecting-lollipops-smart-lock.html>, accessed Feb. 2016. Citado na página 44.

EMVCo. 1994. <https://www.emvco.com/>. Citado na página 44.

FEBRABAN. Pesquisa febraban de tecnologia bancária 2013. In: FEBRABAN. *Congresso e Exposição de Tecnologia da Informação das Instituições Financeiras*. [S.l.], 2013. Citado na página 14.

FIDO Alliance. *FIDO NFC Protocol Specification v1.0*. 2015. Implementation Draft. Citado na página 43.

FIDO Alliance. *Universal 2nd Factor (U2F) Overview*. 2015. Proposed Standard. Citado na página 43.

FIELDING, R. *Architectural Styles and the Design of Network-based Software Architectures*. Tese (Doutorado) — University of California, 2000. Citado na página 40.

HAO, F.; RYAN, P. J-pake: authenticated key exchange without pki. In: *Transactions on computational science XI*. [S.l.]: Springer, 2010. p. 192–206. Citado na página 40.

HOOK, D. *Beginning cryptography with Java*. [S.l.]: John Wiley & Sons, 2005. Citado na página 38.

HORSCH, M.; BRAUN, J.; WIESMAIER, A. *Mobile eID application for the German identity card*. [S.l.], 2011. <http://www.cdc.informatik.tu-darmstadt.de/reports/TR/Mobile_eID_app_for_the_German_ID_card.pdf>. Citado na página 43.

HWMASTER. *Pagamenti NFC con postemobile*. 2013. <http://www.hwmaster.com/21708-pagamenti-nfc-con-postemobile/>. Accessed: 2015-04-27. Citado na página 18.

HöLZL, M. et al. A password-authenticated secure channel for App to Java Card applet communication. *International Journal of Pervasive Computing and Communications*, Emerald Group Publishing Limited, v. 11, n. 4, p. 374–397, nov. 2015. ISSN 1742-7371. <http://dx.doi.org/10.1108/IJPCC-09-2015-0032>. Citado na página 43.

Inside Secure. *VaultIC150/150D/152*. 2015. <http://www.insidesecure.com/Products-Technologies/Secure-Solutions/VaultIC150-150D-152>, accessed Feb. 2016. Citado na página 44.

KOBLITZ, N. Elliptic curves criptosystem. *Mathematics of Computation, vol. 48*, 1987. Citado na página 35.

KOMATINENI, S.; MACLEAN, D.; HASHIMI, S. *Pro Android 3*. [S.l.]: Apress, 2011. Citado na página 13.

LANCRENON, J.; ŠKROBOT, M.; TANG, Q. Two more efficient variants of the j-pake protocol. In: SPRINGER. *International Conference on Applied Cryptography and Network Security*. [S.l.], 2016. p. 58–76. Citado na página 40.

LEóN-COCA, J. M. et al. Authentication Systems Using ID Cards over NFC Links: The Spanish Experience Using DNIe. In: *The 4th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN-2013) and the 3rd International Conference on Current and Future Trends of Information and Communication Technologies in Healthcare (ICTH)*. [S.l.]: Elsevier, 2013, (Procedia Computer Science, v. 21). p. 91–98. <http://dx.doi.org/10.1016/j.procs.2013.09.014>. Citado na página 43.

MATHUR, M.; KESARWANI, A. Comparison between des , 3des, rc2 , rc6 , blowfish and aes. In: *Proceedings of National Conference on New Horizons in IT* -. [S.l.: s.n.], 2013. v. 3. Citado na página 31.

MILLER, V. S. Use of elliptic curves in cryptography. *Advances in Cryptology - CRYPT0 ' 8 5 , LNCS 218, pp. 417-426*, 1986. Citado na página 35.

M'RAIHI, D. et al. *HOTP: An HMAC-Based One-Time Password Algorithm*. [S.l.], 2005. 37 p. <https://tools.ietf.org/html/rfc4226>. Citado na página 51.

MURDOCH, S. J. et al. Chip and PIN is Broken. In: *Proceedings of the 2010 IEEE Symposium on Security and Privacy*. [S.l.]: IEEE, 2010. p. 433–446. <http://dx.doi.org/10.1109/SP.2010.33>. Citado na página 21.

NFC Forum. 2004. <http://nfc-forum.org/>. Citado na página 17.

NFC FORUM. *NFC Data Exchange Format (NDEF)*: Technical specification. [S.l.], 2006. Citado 2 vezes nas páginas 7 e 26.

NFC Forum. *Type 4 Tag Operation, ver. 2.0*. 2011. Technical Specification. Citado 2 vezes nas páginas 19 e 53.

NFC Forum. *Type 1 Tag Operation, ver. 1.2*. 2014. Technical Specification. Citado na página 19.

NFC Forum. *Type 2 Tag Operation, ver. 1.2*. 2014. Technical Specification. Citado na página 19.

NFC Forum. *Type 3 Tag Operation, ver. 1.2*. 2014. Technical Specification. Citado na página 19.

NFC Forum. *Type 5 Tag Operation, ver. 1.0*. 2015. Technical Specification. Citado na página 19.

nfc-tag.de. 2016. <https://www.nfc-tag.de/>. Acesso em: 20 mar 2016. Citado na página 65.

NOHL, K. et al. Reverse-engineering a Cryptographic RFID Tag. In: *Proceedings of the 17th USENIX Security Symposium*. [S.l.]: USENIX Association, 2008. p. 185–193. <http://www.usenix.org/events/sec08/tech/full_papers/nohl/nohl.pdf>. Citado na página 35.

NOHL, K.; PLOTZ, H. Mifare: Little security, despite obscurity. In: *Chaos Communication Congress*. [S.l.: s.n.], 2007. Citado na página 20.

NSA. *Case for Elliptic Curve Cryptography*. 2009. Disponível em: <http://www.nsa.gov/business/programs/elliptic_curve.shtml>. Acesso em: 27 jun 2015. Citado na página 34.

NXP. *MIFARE DESFire EV1 contactless multi-application IC*. 2015. Application Note. Citado 2 vezes nas páginas 7 e 21.

ORGANIZATION, I. S. *International standard ISO/IEC 14443*. 2003. Technical Specification. Citado na página 18.

OSWALD, D.; PAAR, C. Breaking Mifare DESFire MF3ICD40: Power Analysis and Templates in the Real World. In: *Cryptographic Hardware and Embedded Systems – CHES 2011: 13th International Workshop, Nara, Japan, September 28 – October 1, 2011. Proceedings*. [S.l.]: Springer, 2011, (LNCS, v. 6917). p. 207–222. <http://dx.doi.org/10.1007/978-3-642-23951-9_14>. Citado 2 vezes nas páginas 20 e 62.

ROGAWAY, P. Evaluation of some blockcipher modes of operation. *Cryptography Research and Evaluation Committees (CRYPTREC) for the Government of Japan*, 2011. Citado na página 32.

ROSATI, T.; ZAVERUCHA, G. Elliptic curve certificates and signatures for nfc signature records. *Research In Motion, Certicom Research*, 2011. Citado na página 34.

SAHAA, M.; Roy Chowdhurya, D. Provably secure key establishment protocol using one-way functions. *Journal of Discrete Mathematical Sciences and Cryptography*, v. 12, n. 2, p. 139–158, 2009. ISSN 1742-7371. <http://dx.doi.org/10.1080/09720529.2009.10698224>. Citado na página 39.

SHIN, S.; KOBARA, K.; IMAI, H. Leakage-Resilient Authenticated Key Establishment Protocols. In: *Advances in Cryptology - ASIACRYPT 2003: 9th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, November 30 – December 4, 2003. Proceedings*. [S.l.]: Springer, 2003, (LNCS, v. 2894). p. 155–172. <http://dx.doi.org/10.1007/978-3-540-40061-5_10>. Citado na página 39.

SPTRANS. *Bilhete Unico*. 2013. <http://http://bilheteunico.sptrans.com.br/>. Accessed: 2015-09-30. Citado na página 14.

STALLINGS, W. *Criptografia e segurança de redes*. [S.l.]: Pearson, 2008. Citado 3 vezes nas páginas 7, 32 e 33.

Tagstand. 2016. <https://nfctags.tagstand.com/>. Acesso em: 20 mar 2016. Citado na página 65.

TAN, W. H. *Practical Attacks on the MIFARE Classic*. Dissertação (Mestrado) — Imperial College London, 2009. Citado na página 20.

URIEN, P. Introducing TLS-PSK authentication for EMV devices. In: *Proceedings of the 2010 International Symposium on Collaborative Technologies and Systems (CTS)*. [S.l.]: IEEE, 2010. p. 371–377. <http://dx.doi.org/10.1109/CTS.2010.5478489>. Citado na página 44.

WALLET. *Google Wallet*. 2011. <https://www.google.com/wallet/>. Accessed: 2015-09-30. Citado na página 14.

WEBBER, J.; PARASTATIDIS, S.; ROBINSON, I. *REST in practice*. [S.l.]: O'Reilly Media, 2010. Citado na página 42.

WEBXTOOL. *O mercado mobile no Brasil*. 2012. Disponível em: <http://webxtool.com/pt/infograficos/mercado-mobile-no-brasil>. Acesso em: 17 nov 2014. Citado na página 13.

WEISER, M. The computer for the 21st century. *Scientific american*, Nature Publishing Group, v. 265, n. 3, p. 94–104, 1991. Citado na página 13.

WU, T. The Secure Remote Password Protocol. In: *Proceedings of the Network and Distributed System Security Symposium, NDSS 1998, San Diego, California, USA*. [S.l.: s.n.], 1998. <http://www.isoc.org/isoc/conferences/ndss/98/wu.pdf>. Citado na página 40.

Yubico. *YubiKey NEO – Premium Strong Two-Factor Authentication for Secure Logins*. <https://www.yubico.com/products/yubikey-hardware/yubikey-neo/>, accessed Feb. 2016. Citado na página 43.