UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

CAIO VINÍCIUS GOMES BALTHAZAR

UMA PROVA DE CONCEITO EM IDENTIDADE ORGANIZACIONAL MÓVEL

QUIXADÁ
2014

CAIO VINÍCIUS GOMES BALTHAZAR

UMA PROVA DE CONCEITO EM IDENTIDADE ORGANIZACIONAL MÓVEL

Trabalho de Conclusão de Curso submetido à Coordenação do Curso Bacharelado em Sistemas de Informação da Universidade Federal do Ceará como requisito parcial para obtenção do grau de Bacharel.

Área de concentração: computação

Orientador Prof. MSc. Marcos Dantas Ortiz

QUIXADÁ

2014

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca do Campus de Quixadá

B158p Balthazar, Caio Vinícius Gomes
Uma prova de conceito em identidade organizacional móvel / Caio Vinícius Gomes Balthazar. – 2013.
87 f. : il. color., enc. ; 30 cm.

Monografia (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Sistemas de Informação, Quixadá, 2013.
Orientação: Prof. MSc Marcos Dantas Ortiz
Área de concentração: Computação

1.Computação móvel 2. Android (recurso eletrônico) 3. Criptografia de dados (computação) I. Título.

CDD 005.1

CAIO VINÍCIUS GOMES BALTHAZAR

UMA PROVA DE CONCEITO EM IDENTIDADE ORGANIZACIONAL MÓVEL

Trabalho de Conclusão de Curso submetido à Coordenação do Curso Bacharelado em Sistemas de Informação da Universidade Federal do Ceará como requisito parcial para obtenção do grau de Bacharel.

Área de concentração: computação

Aprovado em: _____ / janeiro / 2014.

BANCA EXAMINADORA

_____

Prof. MSc. Marcos Dantas Ortiz (Orientador)

Universidade Federal do Ceará-UFC

_____

Prof. Dr. David Sena Oliveira

Universidade Federal do Ceará-UFC

_____

Prof. MSc. Marcio Espíndola Freire Maia

Universidade Federal do Ceará-UFC

Aos meus pais, Tarciso e Bernadete, que sempre me motivaram a ir mais longe e me deram forças para enfrentar quaisquer desafios.

AGRADECIMENTOS

Agradeço à comunidade de *software* aberto pelas incontáveis horas de esforço e dedicação contribuídas ao ensino e à propagação do conhecimento, sem as quais este trabalho simplesmente não seria possível. Àqueles que alimentam a comunidade com suas dúvidas, experiências e contribuições nos clareiam a mente quando os livros não são capazes.

Agradeço aos coordenadores do curso de Sistemas de Informação, Tânia Pinheiro, Francisco Aragão e Davi Romero, que pacientemente guiaram um estudante inexperiente e um a um foram essenciais na minha jornada. Em nome de Shriram Mohan, Wladimir Tavares, Ricardo Reis, Michael Wollowski, Camilo Almendra, Alex Lo, Críston Souza, Marcos Dantas, Cary Laxer, Carla Moreira, Nadine Shillingford, Andreia Sampaio e Jeandro Bezerra agradeço a todos os brilhantes professores que me apontaram caminhos e serviram de profunda inspiração ao longo desses anos.

Agradeço àqueles que sempre me ouviram, acolheram e aconselharam, me conduzindo pelos caminhos da vida: Eliseuda, Tarciso e Mirtes. Agradeço aos meus pais por todos os sacrifícios feitos ao longo dos anos. Aos meus irmãos, tios e primos. A todos estes, o meu “muito obrigado” pelo apoio incondicional, alegria, exemplo e estímulo.

Em nome de Luana Dias, Rafael Vieira, Caleb Drake, Eduardo Bonet, Geovanny Filho, Samuel Torres, Marcelo Arraes, Franklin Barroso, Thaís Melo, Romário Cabó, Jessica Souza Alves, Regis Melo, Marcelo Barros, André Campos, Rodrigo Matihara, Samuel Freitas, Luís Covatti, Allan Vitor, Daniel de Araújo, Tim Curley, Ricardo Sindeaux e Hector Escobar agradeço a todos os meus amigos, essenciais em todas as etapas da minha vida.

Finalmente, à banca avaliadora, por prontamente aceitar fazer parte desse momento tão especial. Aos companheiros de trabalho, funcionários e desconhecidos que contribuíram sem perceber para que esse sonho se realizasse.

"Tudo parece impossível até que seja feito."
(Nelson Mandela)

RESUMO

Dentre as formas de identificação organizacional comuns, destacam-se o uso de senhas, crachás e dispositivos de acesso especializados. Embora bastante difundidas, elas apresentam algumas ineficiências discutidas no trabalho. Paralelamente, sistemas de segurança modernos caminham para um novo cenário de autenticação conhecido como autenticação de dois fatores. Entretanto, mesmo soluções organizacionais recentes para autenticação de dois fatores frequentemente requerem *hardware* especializado. Este trabalho implementa uma solução para identificação organizacional com autenticação de dois fatores, sem *hardware* especializado, através de uma ferramenta quase ubíqua: *smartphones*. *Smartphones* já são considerados dispositivos multi-propósito devido aos seus recursos e portabilidade. Porém, a facilidade de manipulação desses dispositivos deixa organizações ainda temerosas, mantendo-os potencialmente subutilizados. Um segundo desafio para *smartphones* vem de sua moderada capacidade computacional se comparada aos requerimentos da criptografia assimétrica moderna. Esse desafio se torna ainda mais evidente com a demanda exigente de desempenho por parte dos usuários. O modelo implementado neste trabalho aborda esses e ainda outros desafios. Enquanto trabalhos relacionados utilizam criptografia simétrica, outros canais de comunicação, requerem *hardware* ou plataformas especiais, impõem fortes requisitos de implantação, ou simplesmente se orientam a outros cenários, a solução apresentada se destaca ao utilizar avanços computacionais recentes para operar uma infraestrutura de chave pública simples porém completa, orientada especialmente ao ambiente organizacional e à mobilidade. O sistema é implementado seguindo as recomendações do *National Institute of Standards and Technology* e do *Internet Engineering Task Force*, sem com isso abdicar da simplicidade de uso. Tendo como objetivo principal implementar a identificação organizacional móvel com dois fatores através de *smartphones*, a arquitetura é projetada para suportar dois modos de implantação, dois modos de operação e janelas de operação *off-line*, adaptando-se às necessidades organizacionais de forma transparente. A solução busca ainda ser extensível a outras aplicações, almejando promover a autenticação de dois fatores por diversos setores organizacionais. O processo metodológico foi composto de macrociclos e microciclos de desenvolvimento. Cada macrociclo inclui atividades de análise de requisitos, prototipagem, modelagem, validação, planejamento, implementação e *feedback*. Utilizando técnicas de processamento móvel paralelo, criptografia assimétrica, comunicação visual com códigos QR e a plataforma Android a arquitetura foi implementada em um protótipo funcional apresentado no trabalho. É feita uma comparação entre as implementações da arquitetura com RSA e com Curvas Elípticas, com a criptografia elíptica obtendo melhor desempenho mesmo com nível de segurança superior. Independentemente do algoritmo assimétrico utilizado, a arquitetura se mostra resistente a ataques como homem no meio e repetição, trazendo consigo estratégias para o tratamento do caso em que dispositivos são comprometidos.

Palavras chave: Identificação e autenticação móvel. Criptografia de chave pública. *Smartphones* Android. Autenticação de dois fatores. Códigos QR.

ABSTRACT

Among the most common forms of organizational identification, the use of passwords, ids and specialized access devices stand out. Although quite widespread, they present some inefficiencies discussed is this paper. Simultaneously, modern security systems reach for a new authentication model known as two-factor authentication. Nevertheless, even recent organizational solutions for two-factor authentication often require specialized hardware. This work offers a solution for organizational identification with two-factor authentication without specialized hardware, through an almost ubiquitous tool: smartphones. Smartphones are already considered multi-purpose devices because of their resourcefulness and portability. However, the ease of manipulation of such devices concerns organizations, keeping them potentially underused. A second challenge for smartphones relates to their moderate computational power when compared to the requirements of modern asymmetric cryptography. This challenge becomes even more evident with the high demands of performance from users. The architecture implemented in this paper addresses these challenges and few others. While related works use symmetric cryptography, other communication channels, require special hardware platforms, enforce strong deployment requirements, or simply target other scenarios, the solution presented here stands out by using the latest technology to operate a simple and complete public key infrastructure, specifically designed for the organizational environment and for mobility. The system is implemented following the recommendations of the National Institute of Standards and Technology and the Internet Engineering Task Force, without thereby giving up the simplicity of use. Its main goal is to implement mobile organizational identification with two-factor authentication through smartphones. The architecture is designed to support two deployment modes, two modes of operation of off-line time windows, adapting to organizational needs in a transparent manner. The solution aims to be further extended by other applications, striving to bring two-factor authentication to many organizational sectors. The methodological process was composed of macrocycles and microcycles of development. Each macrocycle includes requirement analysis, prototyping, modeling, validation, planning, implementation and feedback activities. Using mobile parallel processing techniques, asymmetric encryption, visual communication with QR codes and the latest advancements of the Android platform, the architecture was successfully implemented into a working prototype. A comparison between the architecture implementations using RSA and Elliptic Curves is performed, with elliptic encryption getting better performance even with higher security levels. Regardless of the asymmetric algorithm used, the architecture is resistant to attacks such as man in the middle and replay, carrying strategies for dealing with the case in which devices are compromised.

Keywords: Mobile identification and authentication. Public key cryptography. Android smartphones. Two-factor authentication. QR codes.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

AES	*Advanced Encryption Standard*
API	*Application Programming Interface*
CMS	*Cryptographic Message Syntax*
CRL	*Certificate Revocation List*
DSA	*Digital Signature Algorithm*
ECC	*Elliptic Curve Cryptography*
ECDSA	*Elliptic Curve Digital Signature Algorithm*
GPS	*Global Positioning System*
HTTP	*Hypertext Transfer Protocol*
HTTPS	*Hypertext Transfer Protocol Secure*
ICP	Infraestrutura de Chave Pública
IP	*Internet Protocol*
IETF	*Internet Engineering Task Force*
JSON	*JavaScript Object Notation*
MITM	*Man In The Middle*
NIST	*National Institute of Standards and Technology*
OTP	*One Time Password*
PBE	*Password-Based Encryption*
PBKDF2	*Password-Based Key Derivation Function 2*
PIN	*Personal Identification Number*
PKCS	*Public-Key Cryptography Standards*
PRNG	*Pseudo-Random Number Generator*
QA	*Quality Assurance*
REST	*Representational state transfer*

RFC	*Request For Comments*
RSAS	*RSA Security*
SSL/TLS	*Secure Sockets Layer / Transport Layer Security*
TFA	*Two-Factor Authentication*
URL	*Uniform Resource Locator*

SUMÁRIO

1 INTRODUÇÃO

Identificação organizacional, referindo-se ao método pelo qual um indivíduo alega e comprova sua identidade vinculada a uma organização, é um passo de grande importância em contextos organizacionais seguros. Em algumas instituições, é a base para o consumo de uma diversidade de serviços. Quando se mostra necessária, tem sido implementada de diversas formas diferentes. Dentre as mais difundidas, encontram-se o uso de senhas, crachás e dispositivos de acesso especializados. Embora essas as formas de identificação tenham tradicionalmente cumprido seu papel, elas apresentam algumas ineficiências discutidas a seguir.

O uso isolado de senhas, por exemplo, embora muito adequado ao meio virtual, se apresenta consideravelmente inapto ao cenário da identificação organizacional. Essa forma de autenticação demandaria a presença de terminais de verificação espalhados por toda a organização, constantemente conectados. Esses terminais, sejam eles computadores comuns ou dispositivos especializados, apresentam custos de implantação e de manutenção para serem mantidos em rede constante com bancos de dados. Adicionalmente, senhas utilizadas dessa forma se mostram vulneráveis ao mau gerenciamento e a ataques como *phishing* e *malware*, como apontam Hallsteinsen, Jorstad e Thanh (2007), Thanh et al. (2009), Alzomai e Jøsang (2010), Choi et al. (2011), Tanvi, Sonal e Kumar (2011).

O uso de crachás, apesar de não exigir *hardware* custoso, pode não ser adequado ao nível de segurança necessário a um contexto em particular, por depender exclusivamente do reconhecimento humano e da incapacidade alheia de forjar um apetrecho similar. Além disso, não é incomum que sua conferência deixe de ser precisa com o passar do tempo e com o envelhecimento desses acessórios.

Dispositivos especializados de acesso, por outro lado, guardam as credenciais de usuários em seu *hardware*, como em Busold *et al*. (2013). Geralmente são fabricados em formato de cartão, sendo magnéticos ou não. O fluxo de uso desses dispositivos envolve sua apresentação junto a sensores capazes de extrair essas informações e verificar sua autenticidade. No entanto, o uso de *hardware* específico impõe maior custo de implantação e manutenção.

Paralelamente, sistemas de segurança modernos caminham para um novo método de identificação, buscando fortalecer os métodos tradicionais através do requerimento de um duplo fator de autenticação. Esse novo modelo é conhecido como autenticação de dois fatores, comumente abreviado para TFA. Nesse modelo, é necessário utilizar conjuntamente ‘algo que você sabe’ e ‘algo que você possui’, explica DeFigueiredo (2011).

Em âmbito organizacional, já existem iniciativas recentes que buscam substituir as formas tradicionais de identificação por autenticação de dois fatores. A exemplo, em US Deparment Of Defense (2006), governos emitem *smartcards* contendo chips eletrônicos que guardam as credenciais do usuário e requerem um código de acesso para descriptografar essas credenciais, durante cada tentativa de autenticação. Dessa forma, o cartão só tem valor se for utilizado em conjunto com uma senha.

Em um outro exemplo frequente, a autenticação por dois fatores é implantada através da utilização de *tokens* expiráveis gerados por dispositivos conhecidos por *One Time Password* ou OTP, como explica Lee et al. (2010). Um fluxo comum, como em Cheng (2011) e Choi et al. (2011), exige que um usuário confirme o conteúdo de um *token* exibido no seu dispositivo no exato momento da autenticação que, em conjunto com a sua senha, irá satisfazer a dois fatores.

Apesar dos benefícios sobre as formas tradicionais, essas formas de autenticação apresentam dificuldades em sua implantação, como indicam Cheng (2011) e Abe, Itoh e Takahashi (2007). Algumas das mais impactantes surgem da necessidade de *hardware* específico, impondo custos, baixa escalabilidade e menor praticidade quando se tem mais de um vínculo organizacional. Implementar uma forma de autenticação organizacional prática e sem *hardware* especializado não é um problema facilmente solucionável. Consequentemente, ainda é comum a utilização das formas mais simples de autenticação, sendo geralmente desassociadas, abrindo caminho para o mau gerenciamento de credenciais, ataques e desperdícios de tempo, recursos e esforços.

Aqui, será explorada uma oportunidade de solucionar o problema de autenticação organizacional sem *hardware* especializado e ainda implementar autenticação de dois fatores na identificação organizacional. Mobilidade e extensibilidade serão adicionadas ao conjunto de qualidades desejadas. Para tanto, faz-se uso de uma ferramenta cada vez mais ubíqua: *smartphones*.

Smartphones já são considerados dispositivos multi-propósito devido ao seu relativo baixo custo aliado a sua portabilidade e recursos, como explicam Bhutta, Ghafoor e Sultan (2012). O uso de *smartphones* tem crescido e se integrado a diversas atividades pessoais e organizacionais. Simultaneamente, a qualidade das plataformas móveis de desenvolvimento tem melhorado.

Dessa forma, *smartphones* poderiam ser utilizados como dispositivos autenticadores no controle de acesso a serviços organizacionais. Já existem aplicações que oferecem parcialmente essa capacidade, a exemplo de algumas aplicações bancárias e o Google Authenticator (GOOGLE INC., 2013).

No entanto, ainda existem desafios relacionados ao uso de *smartphones* para identificação organizacional. Devido à facilidade de manipulação desses dispositivos, organizações se encontram temerosas quanto à segurança de credenciais confiadas a *smartphones*, impedindo que eles sejam usados como provedores de identidade e mantendo-os como um acessório de segurança potencialmente subutilizado.

Um segundo desafio para os *smartphones* vem de sua moderada capacidade computacional se comparada aos requerimentos da criptografia assimétrica moderna. Esse desafio se torna ainda mais evidente com a demanda de desempenho cada vez mais exigente por parte dos usuários.

Por fim, há ainda o desafio do equilíbrio entre segurança e simplicidade. Uma solução para identificação organizacional precisa ser capaz de operar de maneira ágil e simples buscando alcançar uma integração bem sucedida nos processos organizacionais e aceitação dos usuários. Entretanto, frequentemente procedimentos de segurança contribuem para o aumento da complexidade de processos internos e diminuição de sua agilidade, o que, segundo Gutmann (2013) acaba resultando na má utilização das ferramentas ou sua total inutilização. Esses e outros desafios são explorados neste trabalho, cujo objetivos são definidos a seguir.

1.1 Objetivos

1.1.1 *Objetivo Geral*

Desenvolver uma solução extensível de *software* para identificação organizacional móvel com autenticação de dois fatores utilizando *smartphones*.

1.1.2 *Objetivos Específicos*

a) Projetar uma arquitetura de sistema para identificação organizacional sustentada por apenas *smartphones* e servidores;

b) Projetar uma infraestrutura de segurança baseada em criptografia assimétrica sob as recomendações NIST e IETF para a arquitetura;

c) Projetar um modelo de autenticação com dois fatores resistente a ataques *Man-In-The-Middle* (MITM) e repetição capaz de suportar janelas de operação sem conexão com internet;

d) Projetar um modelo componentizado de *software* capaz de suportar extensão de funcionalidade e reuso do módulo de verificação;

e) Implementar um módulo de armazenamento de segredos integrado às estratégias de segurança da plataforma Android;

f) Implementar um protótipo funcional da arquitetura e suas entidades.

1.2 Organização do texto

O documento está divido em seis seções. Na primeira seção apresentou-se uma introdução ao tema, fazendo uma contextualização e apresentando a problemática vislumbrada, assim como os objetivos geral e específicos.

Na segunda seção é realizada uma revisão sobre os principais componentes do projeto, incluindo alguns trabalhos relacionados, conceitos de criptografia utilizados, detalhes sobre a plataforma de desenvolvimento móvel e elucidações sobre códigos bidimensionais que serão uma característica notável do sistema.

A terceira seção apresenta os procedimentos metodológicos utilizados no desenvolvimento do trabalho, detalhando a metodologia iterativa empregada e os passos que foram necessários para a obtenção da solução aqui apresentada.

A quarta seção traz o detalhamento dos principais elementos e contribuições do projeto. Esta seção se divide em quatro partes, sendo: (a) uma apresentação dos requisitos considerados; (b) uma descrição da arquitetura e sua infraestrutura de segurança; (c) uma descrição dos fluxos de comunicação e procedimentos e (d) um detalhamento da implementação da arquitetura.

A quinta seção apresenta avaliações de desempenho e discute resultados obtidos com a implementação da arquitetura utilizando os sistemas criptográficos RSA e de Curvas Elípticas. Há ainda uma comparação com trabalhos relacionados

Por fim, a sexta seção traz as considerações finais e trabalhos futuros.

2 REVISÃO BIBLIOGRÁFICA

As subseções a seguir oferecem uma revisão de quatro componentes fundamentais do projeto: trabalhos relacionados, criptografia, Android e códigos QR. Na primeira seção, há um breve resgate histórico dos trabalhos mais influentes a esta contribuição. Na subseção de criptografia, são feitas introduções aos conceitos criptográficos utilizados, seguidas de contextualizações do papel de cada conceito na solução implementada. Na segunda subseção, são detalhados alguns elementos da plataforma de desenvolvimento Android relevantes ao trabalho. A subseção de códigos QR faz uma breve caracterização desse tipo de código e de suas qualidades.

2.1 Trabalhos relacionados

Em 2005, McCune *et al* propõe um modelo de identificação demonstrativa e negociação genérica de chaves dentre dispositivos através de um duplo canal de comunicação: *wireless* e visual. O trabalho de McCune *et al* permite a autenticação entre partes que não compartilham um contexto confiável prévio, em uma abordagem quase oposta à deste trabalho. Embora o trabalho de McCune *et al* não tenha orientação organizacional almejada aqui, seu modelo foi inspirador em diversos aspectos, principalmente em sua utilização do canal visual como meio confiável de comunicação.

Ainda em 2005, Hassinen e Hypponen propõem um modelo de autenticação móvel baseado em ICPs governamentais e no uso de SMS. Embora o modelo não tenha desfrutado das capacidades dos *smartphones* modernos devido à época em que foi proposto, os princípios de ICP integrados no seu modelo influenciaram a maneira com que esse trabalho confia a ICP características de autenticidade, integridade e não repúdio.

Em 2011, Tanvi, Sonarl e Kumar apresentam um modelo de autenticação OTP baseado em senhas geradas por *smartphones*. Embora o modelo aqui apresentado siga a filosofia ICP, o modelo proposto por eles influenciou este trabalho à medida que apresenta assiduamente *smartphones* como substituição para *tokens* de hardware.

Em 2012, Bhutta e Ghafoor estendem o protocolo PACS e demonstram a capacidade que *smartphones* têm de serem meios confiáveis para determinar controle de acesso físico a ambientes.

Em 2013, Busold *et al* constrói um modelo chaves inteligentes para automóveis baseando-se em *smartphones,* NFC e ICP. Desconsiderando suas necessidades de *hardware* especializado, Busold *et al* tem uma abordagem muito mais próxima à deste trabalho do que os outros trabalhos mencionados anteriormente. O modelo construído por ele implementa um protocolo de delegação de acesso, utilizando ICP em conjunto com seu próprio protocolo. O modelo de Busold *et al* teve influência considerável nesse trabalho ao despertar a sensibilidade para as vantagens de ICP dentro de uma hierarquia organizacional.

2.2 Criptografia

Neste trabalho, devido à utilização de *hardware* comum, criptografia é a principal barreira entre um atacante e as informações privadas. Segundo Laureano (2005), criptografia representa um conjunto de técnicas para manter uma informação segura, sendo a maneira mais segura de se enviar informações através de um canal de comunicação.

2.2.1 *Qualidades de segurança necessárias*

Dentre as qualidades de segurança promovidas pela criptografia, as consideradas essenciais para o funcionamento da solução aqui desenvolvida são:

a) confidencialidade - propriedade que limita o acesso à informação tão somente às entidades autorizadas pelo proprietário da informação.
b) integridade - propriedade que garante que a informação manipulada mantenha todas as características originais estabelecidas pelo proprietário da informação.
c) autenticidade - propriedade que garante que a informação é proveniente da fonte anunciada e que não foi alvo de mutações ao longo de um processo.
d) não repúdio - propriedade que garante a impossibilidade de negar a autoria em relação a uma transação anteriormente feita

Essas qualidades são inseridas na arquitetura proposta através de procedimentos e protocolos dependentes de uma infraestrutura de segurança baseada em chave pública.

2.2.2 *Infraestrutura de chave pública*

Laureano (2005) ressalta que a infraestrutura de chave pública ou ICP consegue assegurar confidencialidade, integridade, autenticidade e não-repúdio de uma maneira difícil

de ser fraudada e que se apresenta de forma transparente para o usuário. Estes dois pontos, transparência aliada à forte base técnica de seus mecanismos, denotam o aspecto forte desta tecnologia.

Infraestrutura de chave pública é uma iniciativa para utilização de criptografia que tem como base a emissão de chaves por uma organização tida como confiável pelas partes envolvidas. A natureza característica dessa infraestrutura vem do pareamento crítico, mútuo e único entre uma chave pública e sua chave privada. Diversos algoritmos criptográficos podem ser utilizados para implementar essa infraestrutura, sendo conhecidos como algoritmos de criptografia assimétrica.

Essas emissões são realizadas através de certificados digitais, documentos eletrônicos que ligam uma entidade a sua chave publicamente e são assinados pela organização confiável. Ainda segundo Laureano (2005), uma assinatura digital é o processo pelo qual uma entidade gera uma assinatura de uma mensagem criptografada com sua chave privada, que apenas pode ser descriptografada através do uso de sua chave pública, atestada pelo seu certificado digital. Dessa forma, se o reconhecimento da assinatura com a chave pública for bem sucedido, fica comprovado que o assinante tem domínio da chave privada correspondente e verifica-se sua identidade junto à autenticidade da assinatura.

Em ICP, existe ainda um procedimento chamado revogação. Ele se refere ao ato de retirar ou anular a validade de um certificado digital, indicando publicamente que esse certificado não é mais confiável. Dentre as diversas razões que podem levar uma autoridade a revogar um certificado, destaca-se a situação em que a privacidade da chave privada de uma entidade foi comprometida ou sabidamente posta em risco.

A arquitetura proposta neste trabalho é fortemente baseada em ICP. A cadeia de confiança restrita, a habilidade de gerar e reconhecer assinaturas digitais, o controle do ciclo de vida de certificados e a capacidade de revogação dos mesmos são elementos essenciais para a solução implementada aqui. O RSA foi o primeiro algoritmo a implementar assinatura digital, sendo ainda hoje um dos mais utilizados.

2.2.3 *RSA*

RSA é um dos primeiros e mais utilizados sistemas criptográficos de chave pública, desenvolvido por Ronald Rivest, Adi Shamir e Leonard Adleman em 1977. O algoritmo, que teve sua patente expirada em 2000, tem sua segurança baseada na dificuldade

computacional de se fatorar o produto de dois grandes números primos. Embora pequenos avanços tenham sido feitos no ataque ao problema, como apontam Magalhães e Queiroz (2011), grandes tamanhos de chave para o algoritmo são estipulados de forma a garantir a dificuldade computacional de um ataque. O RSA é um dos sistemas criptográficos avaliados como implementação para a infraestrutura de segurança deste trabalho, o outro envolve criptografia de curvas elípticas.

2.2.4 *Criptografia de Curvas Elípticas*

Criptografia de curvas elípticas, ou ECC, é um sistema criptográfico de chave pública relativamente novo desenvolvido independentemente por Victor Miller e Neal Koblitz em 1985. Sua adoção comercial, no entanto, não se popularizou antes de 2004. ECC tem sua segurança baseada na dificuldade computacional de se resolver o problema do logaritmo discreto em curvas elípticas.

ECC tem ganhado recente popularidade sobre o RSA devido a sua capacidade de apresentar segurança equiparável utilizando chaves significativamente menores e velocidade de processamento na ordem de 10 vezes mais rápido que o RSA (MAGALHÃES; QUEIROZ, 2011). A adoção de ECC nas suítes criptográficas e infraestruturas de segurança em todo o mundo tem crescido a medida que órgãos regulamentadores padronizam especificações para sua implementação e resultados acadêmicos atestam sua segurança teórica no atual estado da arte matemática (BARBOSA, 2003). A figura 1 traz uma comparação entre tamanhos de chave de criptografia simétrica, elíptica e RSA.

Figura 1 - Comparação entre tamanho de chaves em esquemas criptográficos

Modelos Criptográficos		
Simétrico	Elíptico	RSA
80	163	1024
128	256	3072
192	384	7680
256	512	15360

Fonte: Magalhães e Queiroz (2011)

Essas qualidades de segurança, tamanho e desempenho se mostram particularmente favoráveis a dispositivos móveis e embarcados, uma vez que estes

dispositivos tem poder computacional e largura de banda mais limitados. Isso faz de ECC o segundo sistema criptográfico avaliado como implementação para a infraestrutura de segurança deste trabalho.

2.2.5 *Recomendações de instituições*

Por todo o mundo, existem órgãos regulamentadores e empresas de segurança influentes que especificam padrões e protocolos a serem utilizados por sistemas criptográficos. Algumas dessas organizações incluem o *National Institute of Standards and Technology* (NIST), a *Internet Engineering Task Force* (IETF) e a *RSA Security* (RSAS para diferenciar do algoritmo).

A seguir, algumas das recomendações dessas instituições que serão referenciadas ao longo do texto, incluindo tamanhos de chave, algoritmos criptográficos, encapsulamento de mensagens criptográficas e formatos de requisição de certificado:

Figura 2 - Recomendações do NIST para tamanhos de chave e algoritmos de acordo com a época

Date	Minimum of Strength	Symmetric Algorithms	Asymmetric	Discrete Logarithm Key	Discrete Logarithm Group	Elliptic Curve	Hash (A)	Hash (B)
2010 (Legacy)	80	2TDEA*	1024	160	1024	160	SHA-1** SHA-224 SHA-256 SHA-384 SHA-512	SHA-1 SHA-224 SHA-256 SHA-384 SHA-512
2011 - 2030	112	3TDEA	2048	224	2048	224	SHA-224 SHA-256 SHA-384 SHA-512	SHA-1 SHA-224 SHA-256 SHA-384 SHA-512
> 2030	128	AES-128	3072	256	3072	256	SHA-256 SHA-384 SHA-512	SHA-1 SHA-224 SHA-256 SHA-384 SHA-512
>> 2030	192	AES-192	7680	384	7680	384	SHA-384 SHA-512	SHA-224 SHA-256 SHA-384 SHA-512
>>> 2030	256	AES-256	15360	512	15360	512	SHA-512	SHA-256 SHA-384 SHA-512

Fonte: (KEYLENGTH, 2013)

Figura 3 – Trecho introdutório sobre *Cryptographic Message Syntax* (CMS) publicada na RFC5652 pela IETF

The CMS describes an encapsulation syntax for data protection. It supports digital signatures and encryption. The syntax allows multiple encapsulations; one encapsulation envelope can be nested inside another. Likewise, one party can digitally sign some previously encapsulated data. It also allows arbitrary attributes, such as signing time, to be signed along with the message content, and it provides for other attributes such as countersignatures to be associated with a signature.

Fonte: (IETF, 2013)

Figura 4 - Visão geral sobre a especificação do formato de requisição de certificado PKCS#10 encontrado na RFC2986, uma republicação IETF do PKCS#10 v1.7 da RSAS

This document describes syntax for certification requests. A certification request consists of a distinguished name, a public key, and optionally a set of attributes, collectively signed by the entity requesting certification. Certification requests are sent to a certification authority, which transforms the request into an X.509 [9] public-key certificate. (In what form the certification authority returns the newly signed certificate is outside the scope of this document. A PKCS #7 [2] message is one possibility.)

Fonte: (IETF, 2013)

2.2.6 *Ataques à segurança*

Existe uma vasta e crescente diversidade de ataques à segurança de sistemas computacionais. Enquanto uma significativa porção desses ataques pode ser mitigada simplesmente pelo uso correto dos procedimentos de criptografia, outros ataques precisam ser contornados por medidas externas dependentes de cada contexto e análise de riscos. Dois tipos de ataques receberam atenção especial durante o desenvolvimento da arquitetura aqui apresentada: homem no meio, ou MITM, e repetição.

Um ataque do tipo homem no meio, como traduzido por Laureano (2005), é uma forma de ataque em que dados trocados entre duas partes são interceptados e potencialmente manipulados por um atacante sem alertar essas partes. Esse tipo de ataque depende da capacidade do atacante de assumir satisfatoriamente a identidade de pelo menos uma das partes. Já um ataque de repetição se caracteriza quando uma transmissão válida de informação entre duas partes é interceptada e retransmitida por um atacante para fins mal-intencionados.

Esses ataques são levados em especial consideração durante o desenvolvimento deste projeto, isto é, tanto na comunicação via rede como no modelo de autenticação por

códigos bidimensionais. A principal ferramenta para proteção da comunicação via rede é o protocolo *Hypertext Transfer Protocol Secure* (HTTPS).

2.2.7 *HTTPS e SSL/TLS*

O protocolo *Transport Layer Security* (TLS), especificado pela IETF baseando-se no seu precedente *Secure Sockets Layer* (SSL), fornece segurança criptográfica a comunicações realizadas através da *Internet*. O protocolo permite que aplicações cliente/servidor se comuniquem de uma maneira projetada para evitar a espionagem, adulteração ou falsificação de mensagens (IETF, 2013). O protocolo SSL/TLS é um componente essencial do protocolo de comunicação HTTPS.

Dado que uma parte ‘S’ ofereça capacidades HTTPS para comunicação com a parte ‘C’, apresente um certificado digital reconhecido por ‘C’ e tenha a posse das chaves privadas correspondentes, então ‘C’ tem a oportunidade de utilizar HTTPS para autenticar ‘S’ e trocar informações criptografadas, sem estar vulnerável aos ataques MITM e repetição descritos na seção anterior. Adicionalmente, caso ‘S’ necessite autenticar a parte ‘C’ através do mesmo protocolo, ‘C’ precisa ter posse de um certificado digital reconhecido por ‘S’ e posse das chaves privadas correspondentes (IETF, 2013).

O HTTPS com SSL/TLS é o único protocolo de comunicação via rede aceito no sistema aqui implementado. Para assegurar que esse protocolo seja sempre utilizado, todas as entidades tiveram suas capacidades de se comunicar por HTTP comum desabilitadas.

2.3 Android

Smartphones atuais dispõem de diferentes plataformas de desenvolvimento. Elas são impulsionadas por múltiplas empresas dos ramos de telecomunicações, *software*, entretenimento e eletrônicos. A partir do final de 2010, a plataforma de desenvolvimento móvel mais popular passou a ser o Android, como atesta Canalys (2011). O Android é um sistema operacional e plataforma de desenvolvimento de código aberto, atualmente mantido pela Google (GOOGLE INC., 2013).

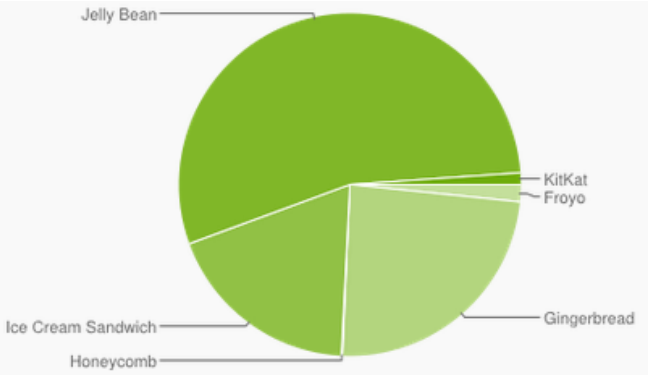
Essa plataforma apresenta características importantes ao contexto desse trabalho que não são encontradas, simultaneamente, nas outras plataformas: sua dominante popularidade no mercado, sua abertura de código sob o *Android Open Source Project*, seu

suporte a uma grande variedade de dispositivos, sua API interna para comunicação entre aplicações e seu suporte integrado ao NFC.

Em 2014, mais de 55% dos dispositivos com Android operam sobre a versão *Jelly Bean* ou superior, internamente conhecida como as APIs 16 a 18 (GOOGLE INC, 2013). A figura 5 traz um detalhamento das distribuições Android em dezembro de 2013.

Figura 5 - Distribuições Android em dezembro de 2013

Version	Codename	API	Distribution
2.2	Froyo	8	1.6%
2.3.3 - 2.3.7	Gingerbread	10	24.1%
3.2	Honeycomb	13	0.1%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	18.6%
4.1.x	Jelly Bean	16	37.4%
4.2.x		17	12.9%
4.3		18	4.2%
4.4	KitKat	19	1.1%



Fonte: (GOOGLE INC, 2013)

Na versão *Jelly Bean*, foi introduzido um conjunto significativo de melhorias de segurança na plataforma, incluindo encriptação de aplicativos e o suporte à proteção de chaves criptográficas através de *hardware*. Um elemento seguro embutido nos aparelhos mais modernos confere essa capacidade. No entanto, a utilização de segurança protegida por *hardware* depende da presença desse elemento físico no aparelho, sendo transparentemente substituída pela tradicional segurança através de *software* caso o aparelho não seja fabricado com um elemento seguro internamente. A segurança através de *software* se baseia no uso do *keystore* do sistema.

2.3.1 *Keystore nativo*

Assim como demonstra Elenkov (2012), o *keystore* nativo do sistema Android é um componente para armazenamento de chaves e certificados, sendo criptografado com o *Advanced Encryption Standard*, ou AES. A senha que protege essa criptografia é derivada do método de desbloqueio do *smartphone* utilizando *Password-Based Key Derivation Function 2* (PBKDF2) com mais de oito mil iterações. Para utilizar o *keystore*, o usuário precisa cadastrar um PIN ou padrão de desbloqueio no dispositivo, de forma que o *keystore* possa ser

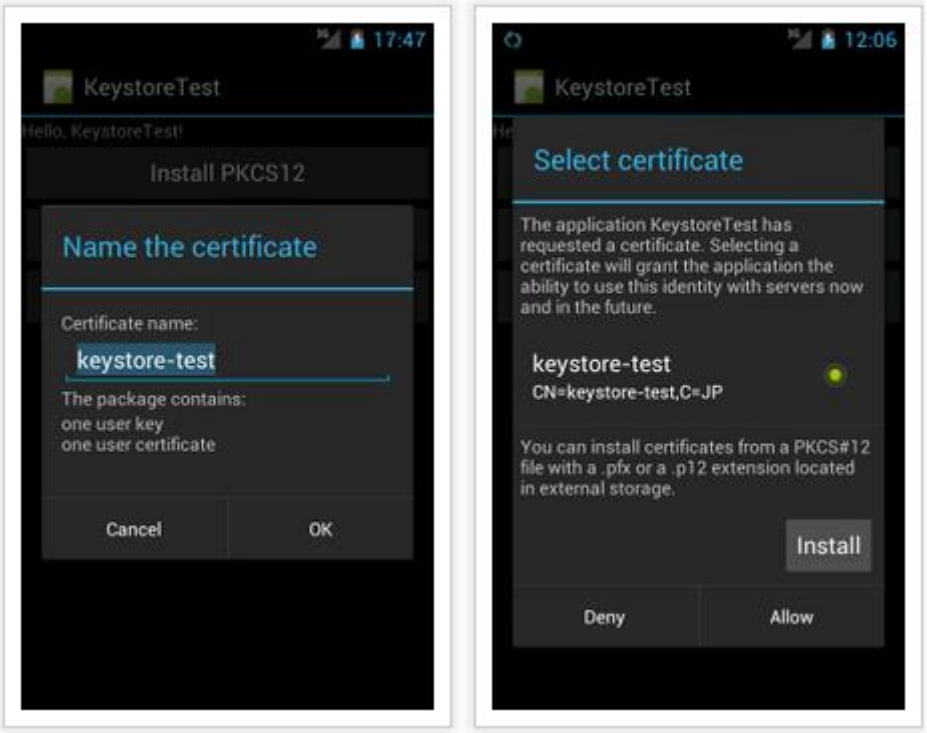
inicializado, protegido e acessado exclusivamente sob permissão do usuário. Uma vez inicializado, o *keystore* só pode ser utilizado por aplicações após o usuário ter desbloqueado o *smartphone*.

Tentativas de remover forçadamente o PIN ou padrão de desbloqueio do dispositivo tornarão o sistema incapaz de derivar a senha do *keystore*, inevitavelmente inutilizando os dados armazenados anteriormente. Adicionalmente, mesmo após o *keystore* ter sido desbloqueado pelo usuário, as chaves contidas nele só podem ser acessadas pela mesma aplicação que as criou, nem mesmo processos do tipo *root* podem ler seus valores em texto plano. Entretanto, embora também não possam acessar os valores em texto plano, processos do tipo *system* podem apagar os textos cifrados do *keystore* para trazê-lo de volta ao estado de fábrica (ELENKOV, 2012). Em *smartphones* com *Jelly Bean* ou superior, que contenham um elemento seguro de *hardware*, o *keystore* do sistema fará uso de proteção de *hardware* transparentemente, operando de maneira ainda mais segura.

Na prática, isso significa que o *keystore* nativo do Android é surpreendentemente seguro para uma solução de *software*: mesmo que um atacante tivesse acesso a um dispositivo como *root* e conseguisse ter acesso ao texto cifrado, ele ainda precisaria da senha do usuário para derivar a chave AES utilizada na encriptação do *keystore*. Experimentar senhas diferentes na tentativa de derivar a chave correta exigiria pelo menos 8.192 iterações de PKBF2 por cada chave tentada, o que é normalmente proibitivamente caro. Além disso, a função de derivação utiliza como "sal" um número aleatório de 128 bits, de modo que tabelas de *hash* previamente-calculadas não podem ser utilizadas.

Apesar de suas qualidades, até a versão 4.3 do Android (API 18), o *keystore* do sistema só estava diretamente acessível às aplicações nativas da plataforma, como as de VPN e WiFi. Antes dessa versão, não havia uma API pública de interação direta com *keystore*, mas apenas através de *intents*. Aplicações eram forçadas a saírem de seu contexto, chamar uma tela do sistema, esperar por interação do usuário e só então receber o resultado dessa interação para dar continuidade às suas tarefas. Esse procedimento, além de ser significativamente lento e intrusivo, abria oportunidade para erros de usuário. A figura 6 traz exemplos dessas telas de interação com usuário antes da versão 4.3.

Figura 6 - Exemplos das telas para uso do *keystore* nativo no Android antes da API 18



Fonte: (ELENKOV, 2011)

2.3.2 *Provedor de criptografia*

O sistema operacional Android executa sobre uma versão modificada da máquina virtual Java chamada Dalvik. O Java Android mantém alta familiaridade com a plataforma Java tradicional, hoje mantida pela Oracle. Não surpreendentemente, a versão Android também delega todas as operações criptográficas a provedores de criptografia através da *Java Cryptography Extension*.

Porém, diferentemente da versão Oracle, a versão Android já inclui uma implementação de criptografia da renomada organização BouncyCastle (LEGION OF THE BOUNCY CASTLE INC, 2013). A implementação criptográfica BouncyCastle é a mais utilizada no mundo Java, tanto por ser aberta quanto por ser australiana, não estando, portanto, submetida às restrições legais americanas de exportação de criptografia.

No entanto, a versão incluída no Android não corresponde integralmente à implementação da original biblioteca, sendo na verdade uma versão diminuída que contém apenas uma subparte de suas funcionalidades. As razões para esse corte não são totalmente

claras à comunidade, a maioria dos desenvolvedores acredita que esteja relacionada ao tamanho da biblioteca e à intenção de integrar nativamente apenas o essencial.

Essa integração nativa se torna um problema à medida que a biblioteca original recebe atualizações mais frequentemente do que o Android é capaz de acompanhar. Desenvolvedores que desejam utilizar versões mais completas da implementação, ou simplesmente mais atuais, não podem utilizar a versão original devido a conflitos com os pacotes já incluídos nativamente. O resultado é que, por algum tempo, desenvolvedores foram forçados a copiar o código da biblioteca em suas aplicações, o que aumentava o risco de erros e o esforço de implementação. Se possível, a alternativa seria utilizar a versão limitada e desatualizada da mesma biblioteca.

Para resolver esse problema, a comunidade aberta criou o projeto SpongyCastle (TYLEY, 2013), que nada mais é do que um empacotamento da implementação original BouncyCastle sob um nome identificador diferente, possibilitando que a biblioteca original seja usada sem conflitar com os pacotes nativos limitados. No protótipo funcional da arquitetura deste trabalho, o empacotamento SpongyCastle foi utilizado como provedor de criptográfico no Android.

2.4 Códigos QR

Ao contrário de códigos unidimensionais, como códigos de barra, onde todas as informações podem ser encontradas em um único corte através do código, códigos bidimensionais utilizam uma segunda dimensão, contendo muitas "linhas" de dados. Eles são comumente utilizados para comunicação visual extremamente rápida no mundo móvel, incluindo, por exemplo, campanhas publicitárias, distribuição de conteúdo e utilizações técnicas.

Entre os códigos bidimensionais mais populares estão o QR e o Data Matrix. Ambos contêm áreas de dados e áreas que ajudam a detectar o código. Adicionalmente, ambos utilizam o sistema de Reed Solomon para recuperação de partes danificadas do código. A figura 7 traz exemplos desses códigos e ressalta suas áreas de controle.

Figura 7 - Exemplos de códigos QR e Data Matrix, ressaltando suas áreas de controle

Fonte: (OREILLY, 2011)

Mesmo sendo mais novo que o Data Matrix, o QR ganhou maior popularidade no contexto móvel devido a sua leitura rápida, estética de personalização e capacidade nativa de suportar caracteres japoneses. O código QR também apresenta algumas vantagens técnicas sobre os códigos Data Matrix que são úteis no contexto deste trabalho, sendo ela maior capacidade máxima e uma taxa de correção de erros configurável (DENSO ADC, 2011).

3 PROCEDIMENTOS METODOLÓGICOS DE DESENVOLVIMENTO

A metodologia de desenvolvimento do trabalho foi derivada pelo autor a partir do modelo tradicional em cascata em conjunto com metodologias ágeis iterativas. Ela emprega uma abordagem algorítmica para a perseguição dos objetivos listados. Os resultados obtidos durante os ciclos de desenvolvimento serão apresentados na próxima seção do trabalho, reservando esta seção para uma descrição da metodologia.

A metodologia é composta de um macrociclo de desenvolvimento que se divide em duas etapas bastante distintas: a cadeia de dependência e o microciclo de desenvolvimento. Essas etapas se alternam constantemente. A figura 11 traz um diagrama de um macrociclo, detalhado no texto a seguir.

3.1 Cadeia de dependência

A cadeia de dependência é a etapa inicial do macrociclo. Seu objetivo principal é criar um modelo minimamente sólido de parte do sistema antes do início da próxima etapa. Sua estrutura busca detectar os problemas conceituais mais básicos. As características marcantes dessa etapa são:

a) é composta por um conjunto ordenado de atividades;

b) ao término de uma atividade, a próxima atividade na sequência deve ser executada, não se permitindo saltos;

c) se, durante a execução de uma atividade, for necessário fazer alterações relacionadas a uma atividade anterior, o fluxo deve ser retornado a essa atividade anterior, com atenção ao item ‘c’. O sequenciamento da cadeia deve ser mantido;

d) ao fim da cadeia, inicia-se a etapa de ciclo de desenvolvimento.

Visivelmente, a etapa de cadeia de dependência se assemelha a uma cascata. Porém, ela é indefinidamente retornável. Mudanças percebidas ao longo da cadeia causam um retorno a uma atividade anterior, seguido de uma conferência de todo o trecho retornado, buscando manter a consistência do modelo projetado. Consequentemente, a ordem das

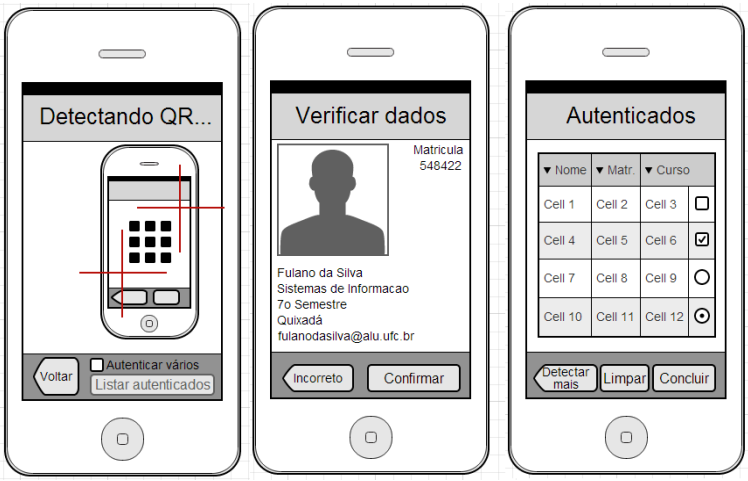
atividades foi estipulada para que mudanças mais fundamentais ressaltassem a necessidade de revisão das atividades seguintes. As atividades desenvolvidas na etapa são agrupadas em fases apenas por mérito de categorização. As fases não têm influência no fluxo da etapa. Ao todo, são 10 atividades dividas em 4 fases.

3.1.1 *Fase 1: Visualização do problema*

Contém 3 atividades. Em ordem:

1. análise dos objetivos:
   - verifica-se se algo precisa ser acrescentado à lista de objetivos. Em seguida, seleciona-se um subgrupo de objetivos a ser trabalhado nesta iteração.
2. esboço parcial da solução (figura 8):
   - faz-se ou incrementa-se um esboço parcial da solução, em sua forma ideal, detalhando as aspirações dos objetivos selecionados.

Figura 8 – Exemplo de esboço primitivo criado durante o projeto



Fonte: elaborado pelo autor

3. especificação dos requisitos:
   - baseando-se no esboço e também nos objetivos, incrementa-se os requisitos formais que devem ser atendidos pela solução após a iteração corrente do macrociclo.

3.1.2 *Fase 2: Projeto e modelagem*

Contém 4 atividades. Ordenadas:

1. identificação de entidades e definição da arquitetura:
   - identifica-se as entidades envolvidas nos requisitos e suas responsabilidades principais, incrementando a arquitetura se necessário.
2. definição da sequência de comunicação e autenticação:
   - sequencia-se os passos principais do fluxo de comunicação. Procura-se mapear de onde a informação se origina e a quem se destina.
3. planejamento de armazenamento:
   - de acordo com os fluxos de comunicação, determina-se aonde, quando e como informações devem ser armazenadas. Ajuda a garantir que os fluxos estejam coerentes.
4. verificação da segurança:
   - assegura-se que seja possível proteger as comunicações e os armazenamentos anteriores.

3.1.3 *Fase 3: Validação do modelo*

Contém apenas uma atividade de verificação: assegurar que os objetivos sejam alcançados pela arquitetura, assegurar que as comunicações e armazenamentos estejam protegidos e assegurar que o modelo de autenticação construído é utilizável.

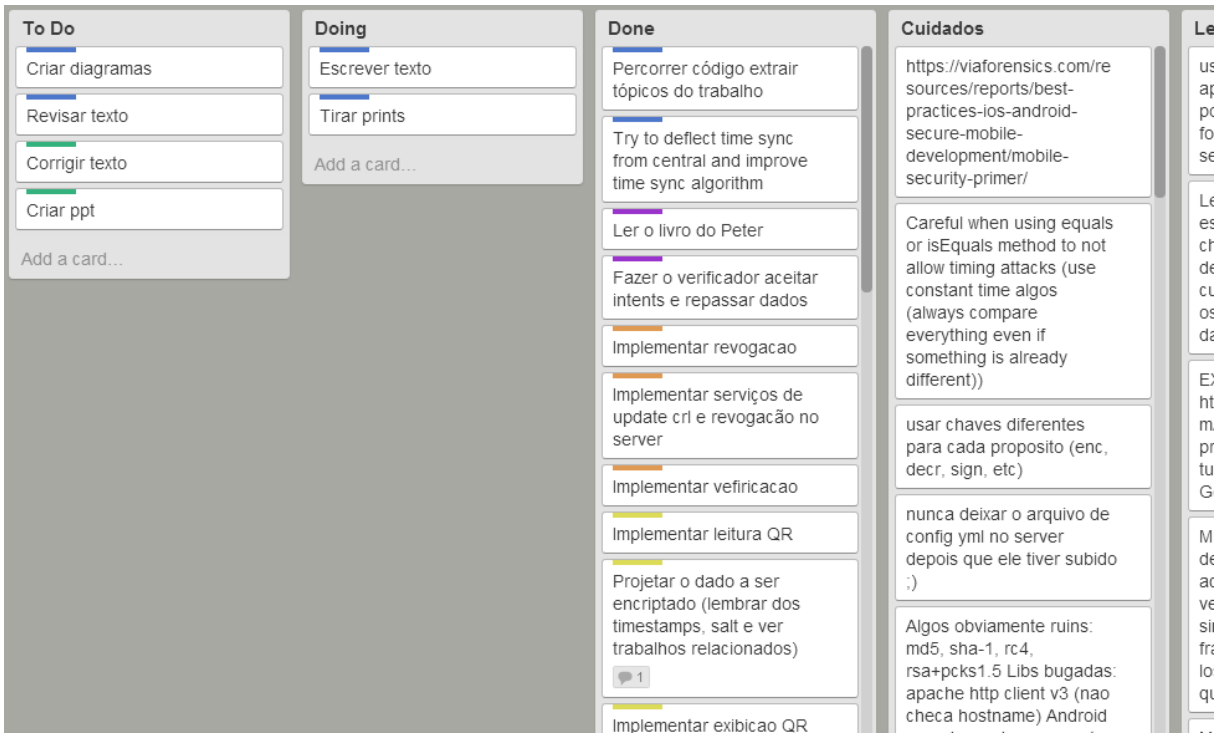
3.1.4 *Fase 4: Planejamento*

Contém 2 atividades. Em ordem:

1. divisão em funcionalidades:
   - quebra-se o modelo validado em múltiplas e pequenas funcionalidades, preparando-o para o planejamento de implementação.
2. ordenação e agendamento (figura 9):

– ordena-se as funcionalidades e faz-se o agendamento das mesmas de acordo com estimativas e tempo de projeto.

Figura 9 – Exemplo simplório de divisão e ordenação através da ferramenta Trello (FOG CREEK SOFTWARE INC, 2013)



Fonte: elaborado pelo autor

3.2 Microciclo de desenvolvimento

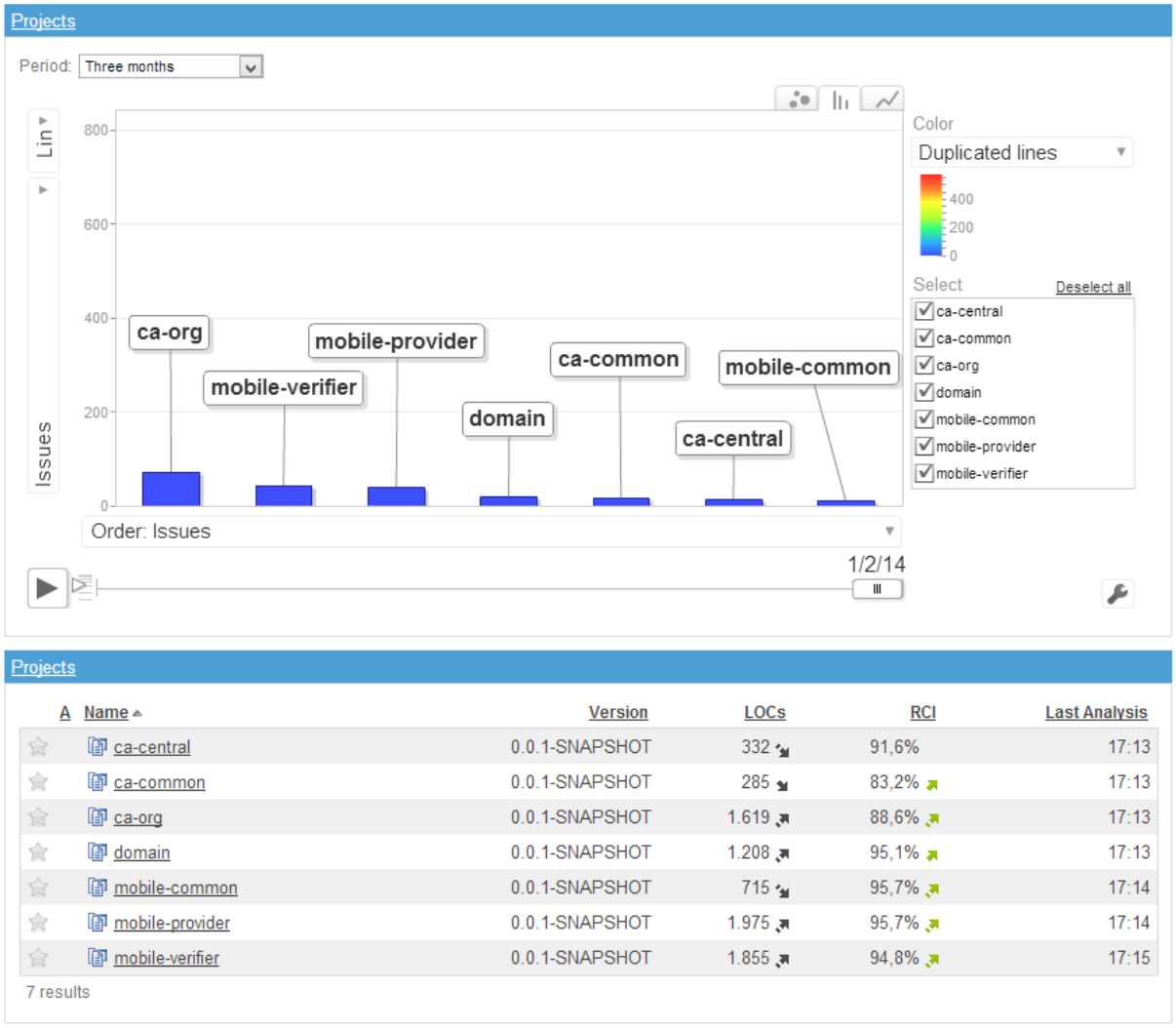
Essa etapa é iniciada sempre que a etapa anterior termina. O microciclo de desenvolvimento tem como objetivo implementar o que foi modelado na cadeia, uma funcionalidade por vez. Sua estrutura é simples e eficaz. Ela se assemelha ao ciclo de desenvolvimento iterativo e incremental. Aqui, funcionalidades planejadas são escolhidas para serem implementadas. O processe segue o fluxo:

1. seleção de funcionalidade;
2. componentização da funcionalidade;
3. implementação, testes e controle de qualidade (figura 10);
4. *feedback* e redirecionamento.

Não raro, essa etapa requer alguma mudança relacionada à etapa anterior. Nesse caso, o redirecionamento será efetuado. Ao final da implementação, podem haver três tipos de

redirecionamento: no pior caso, algo precisa ser mudado e o fluxo será retornado à cadeia de dependência; no caso médio, tudo correu bem e outra funcionalidade pode ser implementada; no melhor caso, todas as funcionalidades foram implementadas e o macrociclo pode reiniciar, selecionando um outro grupo de objetivos. Mesmo no caso de redirecionamento à cadeia de dependência, a mesma eventualmente retornará ao microciclo de desenvolvimento.

Figura 10 - Exemplo de análise de código fonte durante o processo de controle de qualidade com a ferramenta SonarQube



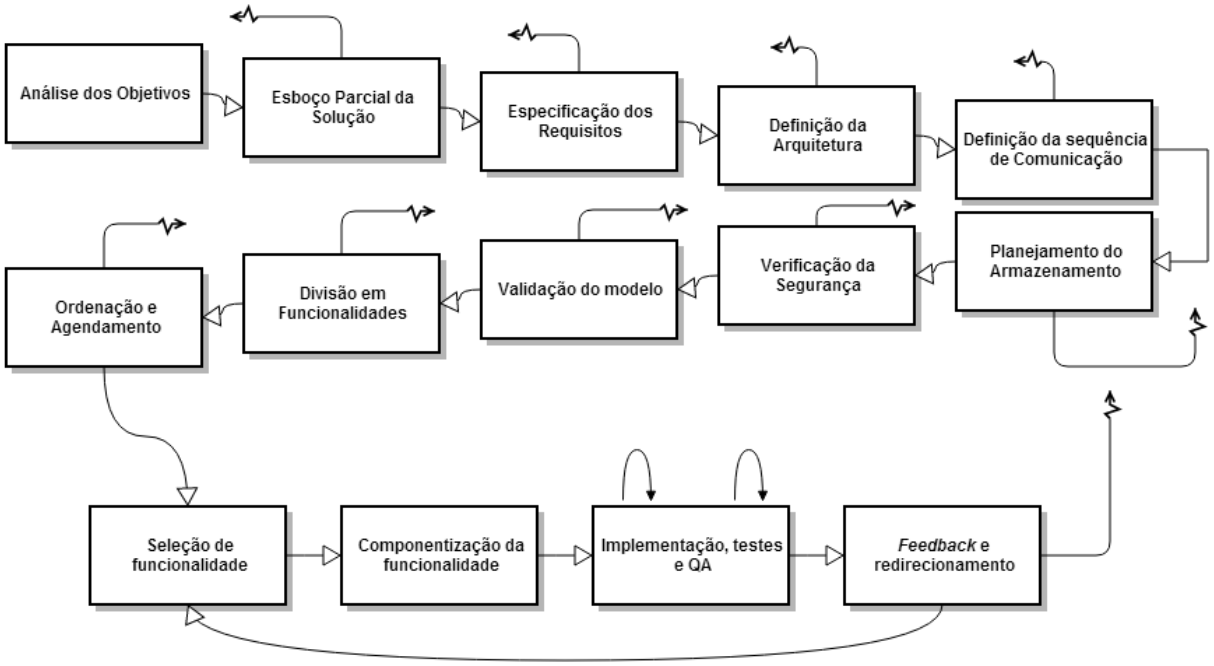
Projects

A	Name	Version	LOCs	RCI	Last Analysis
	ca-central	0.0.1-SNAPSHOT	332	91,6%	17:13
	ca-common	0.0.1-SNAPSHOT	285	83,2%	17:13
	ca-org	0.0.1-SNAPSHOT	1.619	88,6%	17:13
	domain	0.0.1-SNAPSHOT	1.208	95,1%	17:13
	mobile-common	0.0.1-SNAPSHOT	715	95,7%	17:14
	mobile-provider	0.0.1-SNAPSHOT	1.975	95,7%	17:14
	mobile-verifier	0.0.1-SNAPSHOT	1.855	94,8%	17:15

7 results

Fonte: elaborado pelo autor

Figura 11 - Diagrama de um macrociclo metodológico de desenvolvimento



Fonte: elaborado pelo autor

4 IDENTIFICAÇÃO ORGANIZACIONAL MÓVEL

A solução para identificação organizacional móvel será apresentada nas seções a seguir. Suas características são detalhadas ao longo de cinco subseções: requisitos; arquitetura e infraestrutura de segurança; fluxos de comunicação e procedimentos e implementação da arquitetura.

4.1 Requisitos

Os requisitos identificados são derivados dos objetivos específicos na tentativa de se obter um equilíbrio entre usabilidade, desempenho e segurança. Consequentemente, os requisitos principais representam a necessidade de se realizar identificação organizacional através de *smartphones*, abstendo-se de *hardware* especializado, e de integrar autenticação de dois fatores ao processo de identificação.

Para tanto, acrescentam-se requisitos não funcionais de segurança, exigindo que o processo de autenticação apresente qualidades de autenticidade, não repúdio e integridade. Adicionalmente, a confidencialidade dos dados privados das entidades precisa ser assegurada ao armazenar e transferir informações. Ataques do tipo MITM e repetição devem ser mitigados pela solução.

Buscando maior aceitação dentro de processos organizacionais e sempre tentando reduzir a quantidade de pré-requisitos impostos pelo sistema, dois requisitos são ainda considerados no trabalho: a não exigência de cadastros adicionais de usuário e a capacidade de operação durante janelas de tempo *off-line*.

Por fim, há o requisito de extensibilidade, que demanda que a solução seja reusável por outras aplicações de maneira compatível com o padrão de intercomunicação da plataforma Android.

4.2 Arquitetura e infraestrutura de segurança

A geral arquitetura da solução foi em grande parte construída pela integração entre o modelo de autenticação desejado e a infraestrutura de segurança baseada em criptografia de chave pública. As subseções seguintes detalham o modelo de autenticação, o

processo de sincronia temporal desenvolvido, a autenticidade criptográfica dos códigos gerados, as entidades da arquitetura, os seus modos de implantação e os modos de operação.

4.2.1 *Modelo de autenticação*

Uma das características marcantes da solução desenvolvida é sua mobilidade em ambos os lados identificador e identificado. Essas qualidades de mobilidade se somam à abstenção de *hardware* especializado e rapidamente convergem para um modelo de autenticação baseado na interação entre dois *smartphones*.

Um modelo de autenticação baseado em *smartphones* dispõe de poder computacional considerável assim como da oportunidade de implementar a autenticação de dois fatores envolvendo sua posse. Nesse modelo, o *smartphone* em posse do identificado assume o papel de provedor de identidade e o outro *smartphone,* em posse do identificador, assume o papel de verificador. Essas duas partes passarão a ser comumente referenciadas no trabalho como respectivamente provedor e verificador.

O canal de comunicação escolhido para o modelo foi o canal visual. Frente aos canais de comunicação via rede, como *Wi-Fi*, *Bluetooth* e redes de telefonia, o canal visual se destaca por apresentar as seguintes características, adaptadas de McCune *et al* (2005):

a) rapidez de reconhecimento (através de códigos QR) e consequente agilidade da interação;

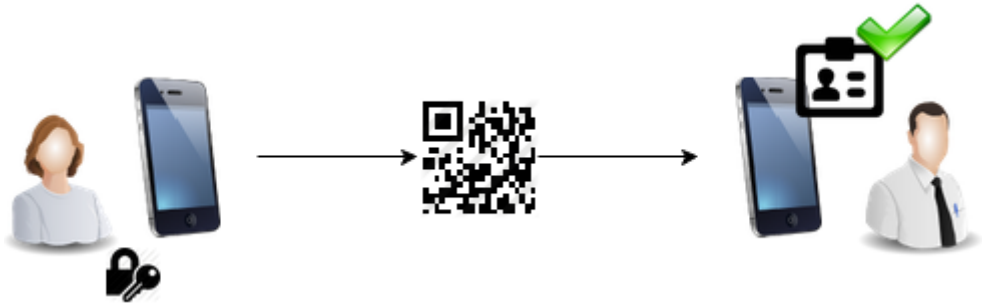
b) eliminação de problemas de conectividade e de identidade em rede através do apontamento ativo e direto das câmeras dos dispositivos;

c) dificuldade de ataque ao canal visual devido aos recursos e esforços necessários serem consideravelmente maiores que em ataques à comunicação via rede;

d) não requer pareamento.

Assim, o modelo de autenticação da solução é construído pela interação visual entre os dispositivos provedor e verificador. Particularmente, o princípio básico do modelo é que o dispositivo provedor deve ser capaz de emitir um código visual de tamanha unicidade que o dispositivo verificador será capaz de extrair sua identidade a partir da interpretação do código. A figura 12 ilustra o princípio básico do modelo.

Figura 12 - Ilustração do modelo básico de autenticação



Fonte: elaborado pelo autor

O uso do canal visual no modelo de autenticação já representa um grande obstáculo a ataques do tipo MITM, como aponta McCune *et al* (2005). Adicionalmente, dado que a informação transmitida pelo código não é confidencial, mas sim única, um ataque do tipo MITM não oferece grandes benefícios a um atacante. Por outro lado, um ataque de repetição poderia ser extremamente danoso ao modelo em sua forma primitiva. Supondo que um atacante fosse capaz de interceptar um código válido, ele poderia repeti-lo e assumir a identidade da vítima indiscriminadamente.

Nesse cenário, a arquitetura é incrementada de forma a mitigar ataques de repetição. Para isso, dois conceitos são introduzidos no modelo de autenticação: sincronia temporal e sessão de autenticação. O princípio de sincronia temporal busca diminuir drasticamente a janela de oportunidade de atacantes ao introduzir uma consciência temporal no código gerado, limitando sua validade a apenas alguns segundos. O princípio de sessão de autenticação estipula que uma identidade só pode ser recebida uma vez por sessão, rejeitando repetições posteriores. Esses dois conceitos, aliados a resiliência do canal visual, reduzem a viabilidade de ataques de repetição a níveis proibitivos.

4.2.2 *Sincronia temporal*

Para que o código assinado possa conter informações temporais, é necessário que haja uma mínima sincronia de relógios entre os dispositivos verificador e provedor. Do contrário, os códigos seriam potencialmente rejeitados como expirados ou adiantados. Embora a sincronia de relógios seja comumente realizada via *Internet*, seu uso como única fonte de sincronia aumentaria as necessidades de conectividade dos dispositivos, limitando a operação *off-line*.

Adicionalmente, visando dificultar a ação de um atacante que obtenha posse temporária de um dispositivo, o tempo utilizado na aplicação deve ser independente do tempo manipulável dos aparelhos. Embora s*martphones* Android disponham de um relógio interno independente do relógio de usuário ele não é suficiente para resolver o problema. Utilizado pelo controle interno de *threads* do sistema, esse relógio conta a quantidade de nanosegundos desde o momento que o aparelho foi iniciado. Enquanto esse relógio é útil para o cálculo de tempo relativo e por não ser manipulável, ele não representa uma informação temporal que possa ser comparada a outro dispositivo. Ele será referenciado a partir de então como relógio *elapsed real time* (ERT).

Com essas necessidades explicitadas, um algoritmo de sincronia temporal foi desenvolvido para os módulos móveis da solução. Esse algoritmo foi inspirado em uma versão simplificada do *Network Time Protocol* (ECE/CIS, 2012), ou NTP, e seu funcionamento é descrito a seguir.

O primeiro aspecto relevante do algoritmo são suas fontes de tempo alternativas. Oportunamente, *smartphones* modernos são capazes de obter tempo a partir de fontes que não dependem de conexão com a internet: redes telefônicas e GPS. O algoritmo desenvolvido utiliza essas fontes transparentemente quando não há conexão disponível, possibilitando a sincronia de tempo mesmo estando "*off-line".* O algoritmo escolhe a fonte mais adequada dentre as opções dependendo do contexto atual do dispositivo.

Figura 13 - Ilustração das fontes alternativas de sincronia temporal

Fonte: elaborado pelo autor

O princípio básico do algoritmo é: tendo sido obtida alguma informação real de tempo, mesmo que seja uma única vez durante a inicialização atual do sistema, é possível calcular o tempo real atual utilizando essa informação obtida e o relógio ERT. Para tanto, no momento do recebimento, cria-se uma estrutura de dados contendo a informação temporal recebida (T) e o estado do relógio ERT (E). A qualquer o momento, o tempo real atual (TR) pode ser calculado a partir do estado do novo estado do relógio ERT (A), seguindo a fórmula:

$$TR = (A - E) + T \quad (1)$$

No entanto, como T é oriundo de uma fonte externa ao sistema como a *Internet*, é necessário compensar os eventuais atrasos na comunicação para mitigar os desvios temporais. O protocolo NTP faz a correção desses desvios utilizando uma complexa estrutura de dados e comunicação hierárquica contendo informações capturadas dos dois lados comunicantes. Dado que na solução implementada aqui se utiliza o mesmo algoritmo para todas as fontes, isto é, *Internet*, GPS e rede de telefonia, o algoritmo NTP original foi simplificado para operar sem as informações que seriam geradas pelo lado servidor.

Assim, seguindo a estratégia de compensação de atraso simétrico do NTP, a fórmula (1) anterior é incrementada da seguinte maneira: registra-se o estado (R1) do relógio ERT ao iniciar a requisição de tempo e registra-se novamente o estado (R2) ao receber a resposta. A qualquer momento futuro, o tempo real aproximado (TRA) será:

$$TRA = ((R2 – R1) / 2) + (A - E) + T \quad (2)$$

Com essa estrutura de dados, tudo que o algoritmo de sincronia precisa fazer é escolher a fonte apropriada de tempo e decidir quando coletar novas amostras. Na implementação da solução, o algoritmo escolhe a fonte de tempo de acordo com conectividade atual do dispositivo:

a) caso haja conexão com a *Internet* disponível, via pacote de dados ou *WiFi*, essa será a fonte utilizada;

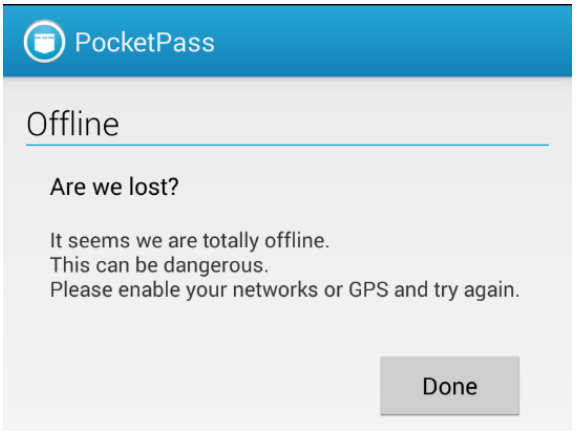
b) caso não haja conexão com a *Internet* disponível, a fonte de tempo será a rede de telefonia;

c) caso a rede de telefonia também não esteja disponível, a fonte de tempo utilizada será o GPS;

d) caso não seja possível contatar a *Internet*, a rede de telefonia, nem mesmo o GPS e nenhuma informação confiável tenha sido capturada previamente, o

usuário é alertado de que o módulo se encontra sem fontes confiáveis de tempo, sendo solicitado a ativar alguma fonte por um breve período.

Figura 14 - Ausência de fontes temporais



Fonte: elaborado pelo autor

Exceto no caso em que todas as fontes estão inacessíveis desde o momento em que a aplicação foi iniciada, a sincronia temporal é automática e transparente ao usuário. Imediatamente ao iniciar a aplicação, enquanto o usuário reage à abertura da primeira tela, o gerenciador de tempo é disparado e age em segundo plano mantendo a sincronia temporal sempre que necessário. Adicionalmente, nenhuma informação temporal é descartada prematuramente ou atualizada sem necessidade, permitindo que trocas de contexto frequentes não exijam novas sincronias em um período de tempo muito curto.

4.2.3 *Autenticidade criptográfica*

Para garantir a autenticidade do código provido na autenticação, assim como sua integridade e não repúdio, faz-se uso de criptografia assimétrica. Um código com essas qualidades é precisamente o que é representado pela assinatura digital em uma ICP.

No contexto da solução, a criptografia assimétrica traz benefícios sobre criptografia simétrica, como a capacidade de autenticar o código provido sem armazenar nenhuma informação crítica no dispositivo verificador. No caso alternativo em que criptografia simétrica fosse utilizada, ambos os dispositivos precisariam conter uma chave compartilhada para realizar a verificação, duplicando o fator de risco no caso de furto ou perda de um dos dispositivos. Além disso, um esquema de atualização de chaves precisaria ser desenvolvido para manter o dispositivo verificador a par de todas as chaves que ele precisa verificar, aumentando a complexidade do projeto e as demandas de conectividade. Uma

alternativa a esse esquema seria uma delegação da verificação simétrica a uma entidade não móvel, mas nesse cenário a conectividade seria um elemento mandatório.

No caso da criptografia assimétrica, dentro de uma ICP, os dados necessários ao dispositivo verificador se resumem ao certificado da organização certificadora e, no caso do modo de operação por revogação, a lista de certificados revogados. Nenhuma dessas informações é confidencial e ambas possuem ciclos de atualização bem definidos. No entanto, ICP exige um conjunto notável de procedimentos de controle, como solicitação de certificados, assinatura dos mesmos, verificações de validade, renovação e revogação. Esses fluxos são de fato abordados pela solução implementada, como descrito na seção sobre procedimentos.

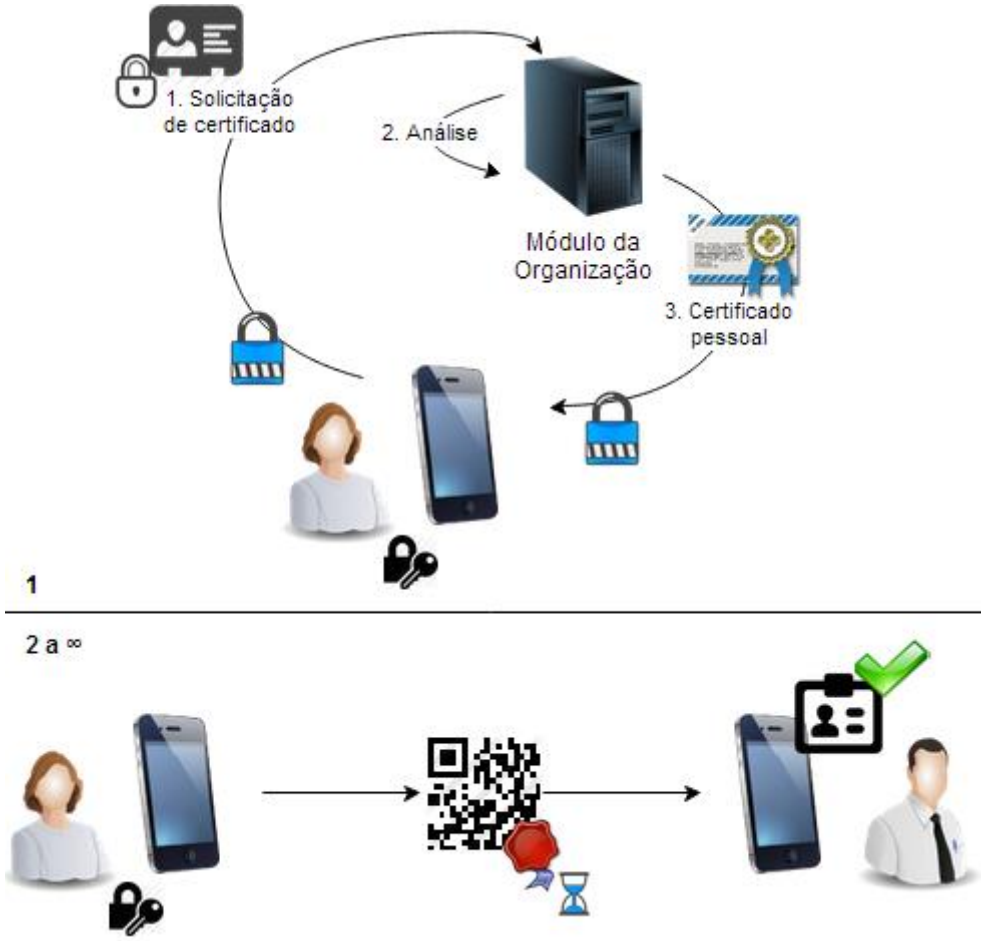
4.2.4 *Entidades da arquitetura*

Para que o dispositivo provedor seja capaz de criar assinaturas visuais que o identifiquem unicamente, ele necessita ter um par de chaves vinculado à sua identidade organizacional por meio de um certificado. Em ICP, certificados são gerados por autoridades certificadoras confiáveis, após uma solicitação.

Nesse contexto, é visivelmente adequado que a própria organização que está implantando a solução e contém os dados de todas as identidades organizacionais exerça o papel de autoridade certificadora dentro dessa ICP privada. Assim, a organização tem poder para analisar solicitações de certificados, controlar o ciclo de vida dos mesmos e liberdade para anexar quaisquer informações aos perfis dos seus usuários vinculados. A estrutura federada de uma ICP se adequa bem à estrutura também hierárquica de uma organização.

Até então, relembrando o modelo de autenticação, três entidades lógicas foram identificadas: uma módulo para *smartphones* provedores de identidade, um módulo para *smartphone*s verificadores e um módulo para que a organização exerça suas funções regulamentadoras, se comunicando com os módulos móveis quando necessário. A figura 15 traz uma simplificação do modelo de autenticação operado sob a arquitetura ICP.

Figura 15 – Exemplo de solicitação de certificado seguida de autenticação, sob perspectiva do aplicativo provedor



Fonte: elaborado pelo autor

A arquitetura projetada aqui vai mais longe e se prepara para suportar múltiplas organizações transparentemente. Nesse pensamento, cada organização representa o topo de uma ICP independente. Para operar nesse cenário, os módulos móveis se mantém, incialmente, independentes de qualquer organização e suas respectivas ICPs.

Para que os módulos móveis sejam capazes de operar sob determinada ICP em um determinado período de tempo, eles precisam receber informações sobre como contatar o módulo da organização que está no topo dessa ICP. A partir do primeiro contato, os módulos móveis operam da mesma maneira que operariam sob uma única ICP, exceto que agora são capazes de alternar seu contexto atualmente ativo, de uma ICP para outra, livremente.

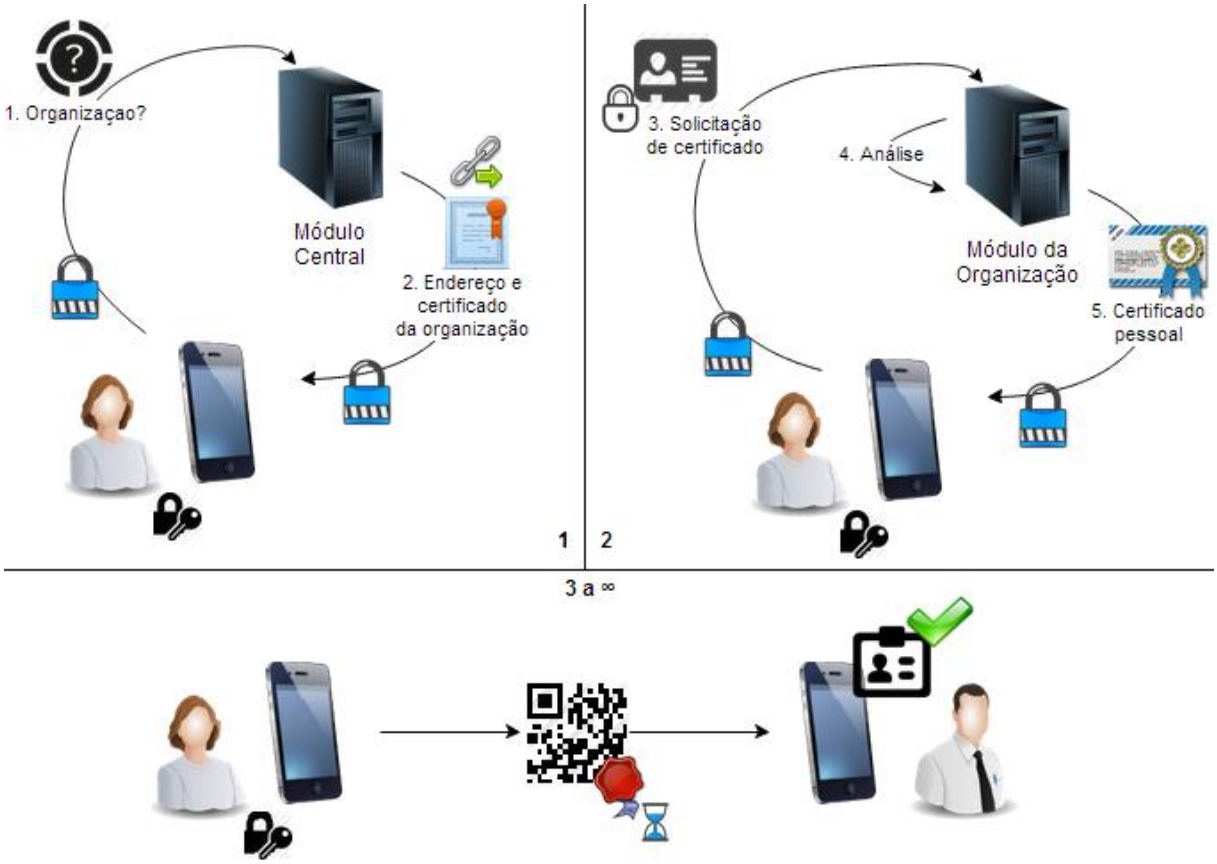
Portanto, uma quarta e última entidade que completa a arquitetura da solução implementada é a central de direcionamento. Esse papel se torna necessário à medida que a arquitetura é construída para comportar múltiplas organizações simultaneamente. O papel da

central de direcionamento é simplesmente fazer registro das múltiplas ICPs organizacionais e fazer o direcionamento de usuários à ICP correta, não exercendo função de certificação.

É importante ressaltar que as comunicações envolvidas nesses protocolos são cruciais e devem ser cuidadosamente protegidas contra ataques MITM e repetição usando HTTPS com verificação de certificados (representado pelo cadeado azul nas ilustrações). A seção de comunicações detalha essa segurança.

Esse papel direcionador permite que os módulos móveis possam alternar livremente entre diferentes ICPs sem ter qualquer alteração em seu código, utilizando a mesma e única instalação. Além disso, o módulo central de direcionamento possibilita o uso comercial da solução proposta. A arquitetura final é então composta por apenas 4 entidades lógicas. A figura 16 traz um exemplo de comunicação nessa arquitetura.

Figura 16 - Exemplo de comunicação na arquitetura multi-organizacional, sob perspectiva do aplicativo provedor



Fonte: elaborado pelo autor

A arquitetura pode ser implantada de duas maneiras: em um modo uni-organizacional ou em modo comercial.

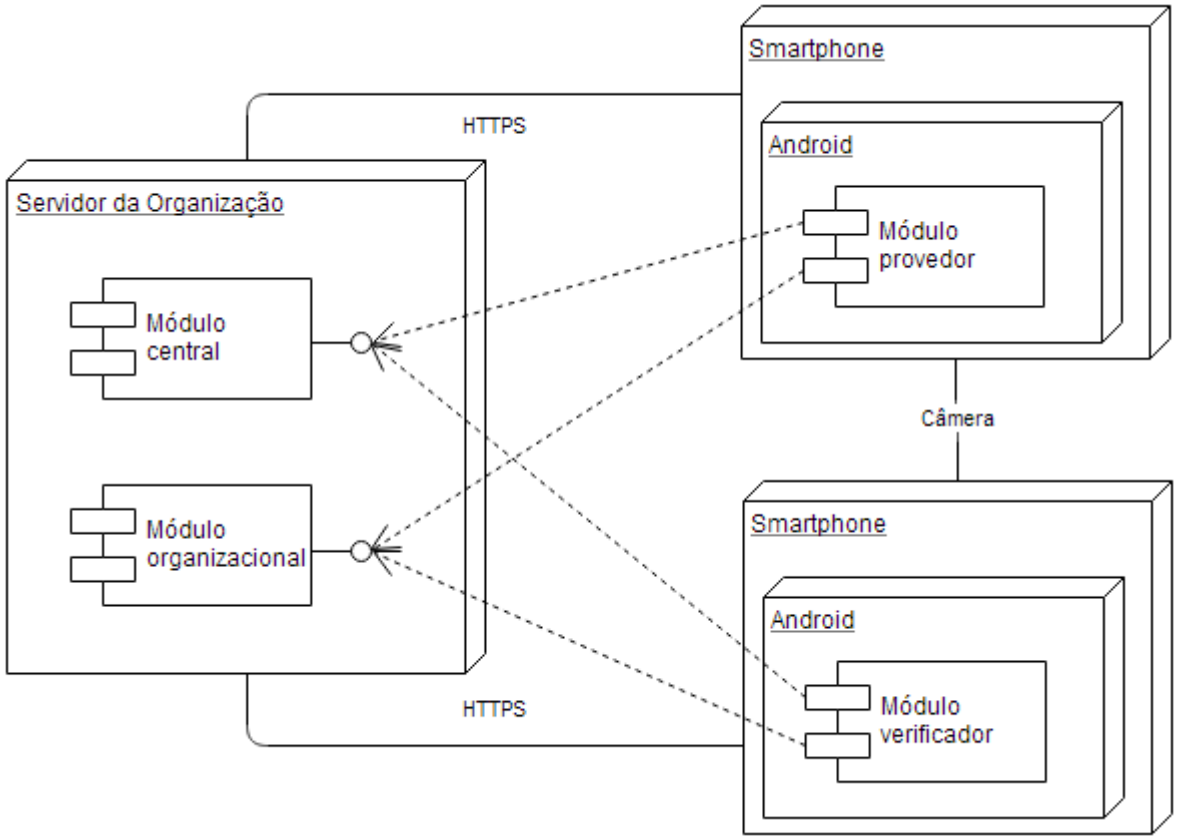
4.2.5 *Modo de implantação uni-organizacional*

Em seu modo de implantação mais simples, a arquitetura assume uma postura intraorganizacional. Ambos os papéis organizacional e de direcionamento são realizados pelos servidores da organização, podendo inclusive serem o mesmo servidor. É importante notar que as 4 entidades lógicas da arquitetura sempre estarão presentes, embora apenas 3 entidades físicas sejam necessárias nesse modo de implantação.

O módulo central é então configurado para fazer apenas um direcionamento: o da própria organização. A arquitetura funcionará transparentemente, uma vez que a busca por informações de organizações não será visível ao usuário provedor de identidade em nenhum dos modos de implantação. O mecanismo de automação dessa requisição está detalhado na seção sobre comunicações.

A organização em questão injeta nos módulos móveis um certificado correspondente ao seu próprio módulo central de direcionamento e os distribui para os seus usuários através de um meio confiável, como as centrais de *software* da plataforma ou através de algum processo definido internamente. Esses usuários poderão então executar o processo de instalação dos aplicativos em seus *smartphones* e fazer uso do fluxo normal de autenticação, detalhado mais à frente.

Figura 17 - Diagrama do modo de implantação uni-organizacional



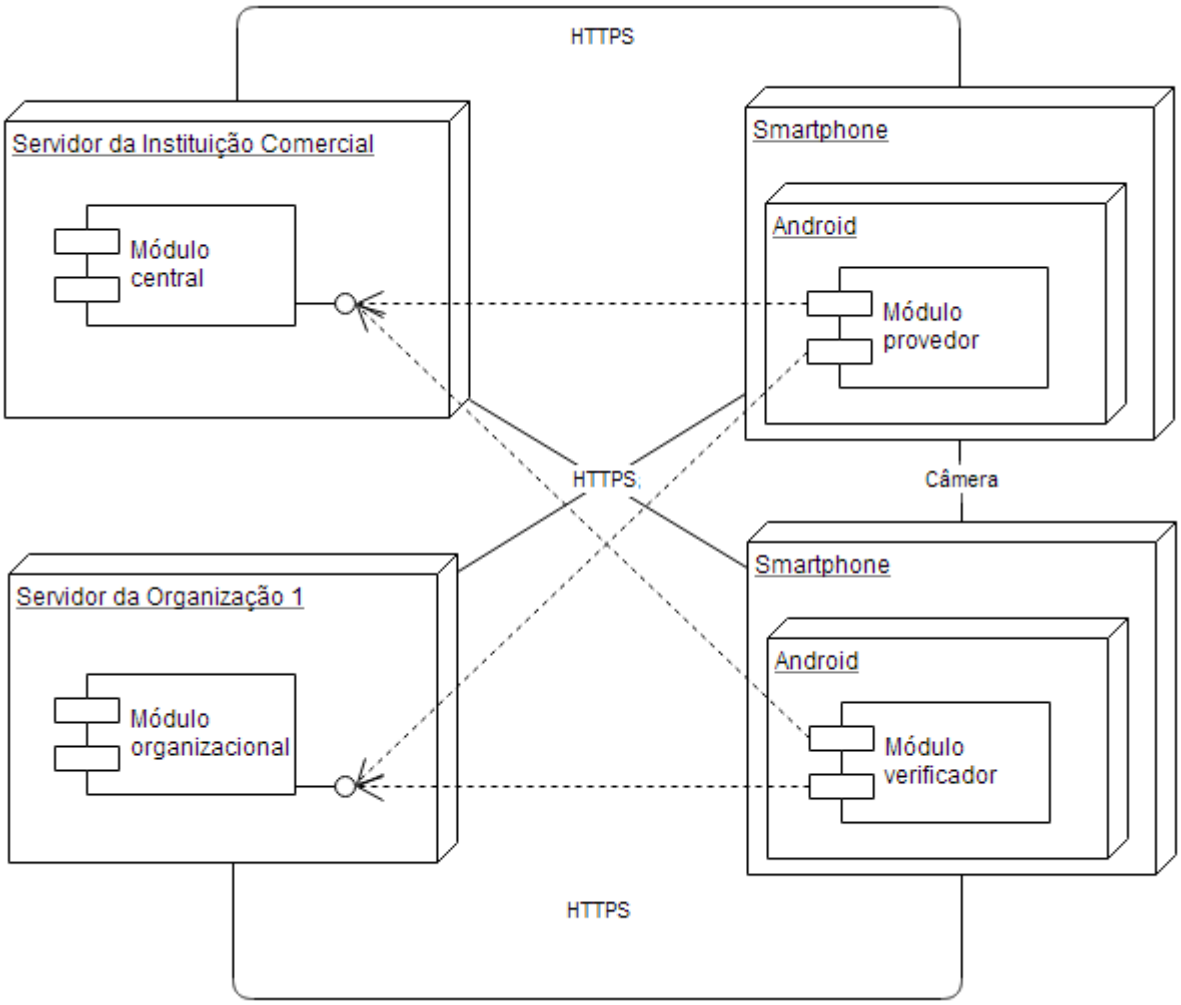
Fonte: elaborado pelo autor

4.2.6 *Modo de implantação comercial ou multi-organizacional*

No modo de implantação multi-organizacional, a arquitetura assume uma postura comercial centralizada, capaz de suportar múltiplas organizações e seus respectivos usuários. O papel de direcionamento é assumido por um servidor de uma instituição comercial. Essa instituição injeta nos aplicativos móveis seu certificado digital e os distribui para os usuários através de um meio confiável, por exemplo, centrais de *software*.

Em seguida, organizações que desejem fazer uso da solução realizam seu registro junto a essa instituição comercial fornecendo, dentre outros dados, seus respectivos certificados. Quando usuários nos módulos móveis necessitarem de certificados para suas identidades, esses módulos irão automaticamente baixar as informações da organização adequada a partir da central e imediatamente abrirão conexões diretas e protegidas com a organização para negociação da solicitação de um novo certificado próprio.

Figura 18 - Diagrama do modo de implantação comercial



Fonte: elaborado pelo autor

4.2.7 *Modo de operação com revogação*

O controle do ciclo de vida de certificados é um ponto crucial em qualquer ICP. A arquitetura implementada aqui é projetada para operar em dois modos distintos. Os detalhes do modo de operação são independentemente controlados por cada organização por meio da configuração do seu módulo organizacional, sendo automaticamente respeitados pelos módulos móveis que venham a se comunicar com ele.

O modelo tradicional de ICP utiliza certificados de longa validade e trata casos excepcionais através do uso de revogação. Neste modo de operação, os certificados emitidos aos usuários por sua organização contêm uma longa data de validade. No caso do comprometimento de algum usuário durante esse período, seu certificado é acrescentado à lista de revogação, que por sua vez tem um curto período de validade.

Nesse modelo, o sucesso do processo de autenticação envolve, dentre outros fatores, a validade do certificado que assinou o código juntamente com sua ausência na lista de revogação. Na solução, o módulo verificador é implementado de forma a controlar as sessões de autenticação de forma compatível com a validade da lista de revogação, ou CRLs. A organização carrega, portanto, a autoridade de controlar a frequência de atualização de suas CRLs e consequentemente a duração máxima das sessões de autenticação nos dispositivos verificadores. A posse de uma CRL não expirada é obrigatória para a operação do módulo verificador, sendo por isso fator determinante no tamanho da janela de operação *off-line* do mesmo.

Esse modo de operação envolve a revogação ativa de certificados comprometidos através de uma funcionalidade disponível no módulo organizacional, que pode ser utilizada pelos próprios usuários através da internet usando suas credenciais organizacionais. No caso de necessidade de utilização dessa funcionalidade, como em todo acesso HTTPS via *browser*, um usuário é recomendado a ficar atendo a ataques de *phishing.*

Alternativamente, uma organização pode optar por um segundo modo de operação: certificados curtos.

4.2.8 *Modo de operação com certificados curtos*

Embora o modo de operação tradicional, baseado em revogação, comtemple satisfatoriamente os requisitos da arquitetura, Gutmann (2013) ressalta que seus pormenores podem parecer agressivos à simplicidade da operação geral do ponto de vista humano. Se este for o caso, uma organização pode optar por um segundo modo de operação implementado na solução: utilização de certificados ultracurtos.

Tomando vantagem do rápido e automatizado processo de requisição de certificados (descrito na seção de comunicação), uma organização pode estipular uma validade ultracurta para seus certificados emitidos e uma validade extralonga para suas CRLs. Nesse cenário, os usuários do módulo provedor fariam renovações frequentes de seus certificados enquanto o módulo verificador seria forçado a atualizar suas CRL com muito menos frequência.

Esse modo de operação, defendido por Gutmann, utiliza-se da validade curta dos certificados como principal fator de segurança, diminuindo-se a utilidade das CRLs. Um dispositivo comprometido teria sua janela de tempo útil ao atacante drasticamente reduzida,

na maioria dos casos tornando desnecessária a revogação dos certificados em questão, dependendo de quão rápida seja a expiração dos mesmos. Como exemplo, uma organização pode optar pela expiração diária de certificados, ou mesmo dentro de horas ou minutos. Dessa forma, acontece uma simplificação do tratamento de casos excepcionais, já que nada precisa ser feito, ao custo de uma atualização mais frequente no caso normal.

É importante notar que a arquitetura nesse modo de operação continua a atualizar CRLs periodicamente, porém não mais de maneira impeditiva, através de um mecanismo não intrusivo de ‘melhor esforço’ executado em segundo plano. Embora a necessidade prática de atualizá-las seja reduzida, a organização pode regular um equilíbrio entre os tempos de atualização de certificados e CRLs ao seu próprio contexto. Os módulos móveis fazem o gerenciamento de certificados e CRLs de maneira independente para cada ICP organizacional sob a qual operam.

Esse modo de operação aumenta consideravelmente a janela de operação *off-line* dos dispositivos verificadores, transferindo parte da demanda de conectividade ao dispositivo provedor.

4.3 Fluxos de comunicação e procedimentos

Nesta seção, são detalhados os fluxos de comunicação entre as entidades durante a execução de diversos processos, juntamente com procedimentos de segurança executados em cada um deles. Antes de adentrar nos procedimentos, é importante capturar um pouco do espírito de desenvolvimento móvel empregado no trabalho.

Uma das técnicas de desenvolvimento utilizadas na implementação desses procedimentos busca driblar parte da percepção do usuário sobre a demanda computacional ou de conectividade envolvidas nos mesmos, através de processamento paralelo. Em ambos os módulos móveis, alguns processamentos independentes são realizados paralelamente à solicitação de informações ao usuário.

Por exemplo, enquanto o usuário está digitando suas credenciais organizacionais para instalar uma nova identidade, o módulo provedor estará adiantando o cálculo sobre pares de chaves em segundo plano para uso futuro. Em um outro exemplo, enquanto o agente verificador está capturando o código QR com sua câmera, o módulo verificador estará fazendo sua sincronia temporal em segundo plano. Esses exemplos são encontrados ao longo

de outros procedimentos independentes que podem ser realizados de forma paralela nos dispositivos móveis.

Outra filosofia de desenvolvimento utilizada no trabalho foi a de que toda operação criptográfica deve ser delegada às bibliotecas e provedores confiáveis. O ato de escrever implementações próprias de algoritmos criptográficos pode levar a introdução de falhas críticas de segurança.

Caso seja necessário, informações sobre desempenho da aplicação podem ser encontrada na seção de comparação entre RSA e ECC.

4.3.1 *Confiança transitiva*

Para que a segurança em rede seja eficiente contra ataques MITM, as comunicações devem ser protegidas com HTTPS autenticado. O protocolo HTTPS pode operar com ambas as partes sendo autenticada, apenas uma das partes ou nenhuma delas. No entanto, no cenário em que nenhuma das partes é autenticada, embora se tenha confidencialidade por meio de criptografia, não se pode assegurar que as partes envolvidas na comunicação são realmente quem dizem ser.

Assim, em ambos os modos de implantação, os módulos móveis já são distribuídos com o certificado do módulo central embutido. Dessa forma, são capazes de autenticar a central durante a busca de informações sobre organizações e assegurar que as respostas de direcionamento são legítimas. A partir de então, os certificados da organização desejada são baixados da central juntamente com o endereço das mesma, permitindo que mesmo a primeira conexão com a organização já seja protegida por HTTPS com verificação de certificado. Uma das suposições do trabalho é a de que a central de direcionamento não se encontra comprometida, o que é uma suposição razoável considerando o seu modo de operação comercial e crucial na arquitetura.

Como mencionado na revisão, as entidades móveis do sistema utilizam clientes de conexão customizados que somente são capazes de realizar conexões seguras, isto é, em que a outra parte está autenticada com certificado reconhecido pelo módulo. Os módulos organizacional e central, por sua vez, apenas respondem a HTTPS.

Acontece portanto uma transferência de confiança, uma vez que no início os módulos móveis confiam exclusivamente na central e gradualmente “aprendem” a confiar nas organizações apontadas pela mesma.

4.3.2 *Inicialização*

Para sua inicialização, o módulo central requer apenas a inicialização do registro de organizações. Um ponto notável do processo de instalação de identidade nos módulos provedores é a maneira automatizada com que as informações da organização desejada são buscadas na central. Essa busca transparente só é possível devido às informações fornecidas pela organização no momento em que ela se registra com a central. Essas informações incluem:

a) uma chave organizacional:
   - uma curta cadeia de caracteres capaz de identificar unicamente aquela organização. É utilizada, dentre outros momentos, ao solicitar uma captura ao módulo verificador.

b) o endereço eletrônico do módulo organizacional;

c) o certificado digital utilizado na comunicação HTTPS do módulo organizacional;

d) uma coleção de domínios de e-mail para inferência automática de vínculo organizacional:
   - a coleção deve conter apenas domínios vinculados à organização e devem ser únicos na central. No modelo comercial, a aceitação ou não do cadastro organizacional fica a livre critério da central.

Assim, durante o processo de instalação de identidade no módulo provedor, basta que o usuário entre com suas credenciais organizacionais e sua respectiva organização será devidamente inferida e contata para fazer a aprovação da sua requisição de certificado. Através da credencial organizacional informada, o módulo central é capaz de inferir a organização responsável por atender àquela demanda.

Por outro lado, a inicialização do módulo organizacional envolve o carregamento da base de dados de usuários vinculados, juntamente às suas informações complementares. As informações básicas utilizadas no protótipo incluem:

a) o login de usuário na forma de e-mail;

b) um *digest* de sua senha;

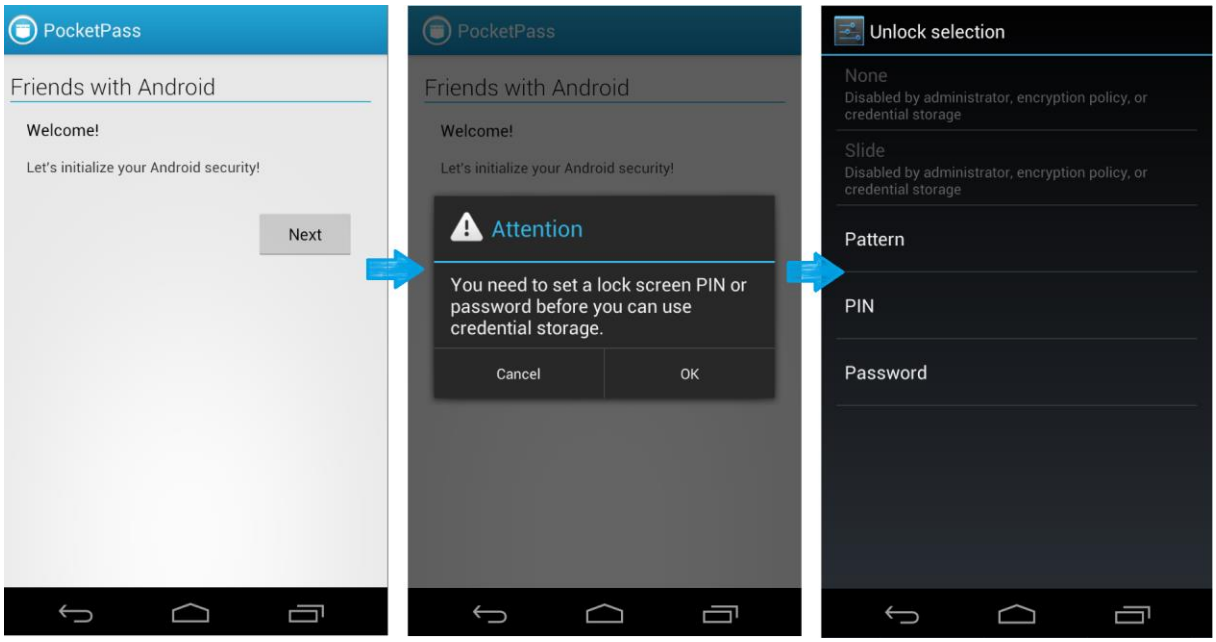
c) um nome identificador único do usuário a ser exibido após o reconhecimento pelo módulo verificador. Pode-se utilizar qualquer sequência reconhecível desejada, como nomes próprios, matrículas, e-mails ou uma combinação desses.

No exemplo do protótipo, as informações complementares utilizadas são fotos.

Dentre as 4 entidades lógicas da arquitetura, a que carrega o maior processo de inicialização é o módulo provedor. Isso se deve à necessidade de inicializar o *keystore* do sistema, como apontado na revisão bibliográfica, e de criar uma identificação da instalação.

Após a primeira execução do aplicativo, o módulo provedor irá verificar se o *keystore* do sistema está acessível. Caso o usuário do *smartphone* já use um PIN ou padrão de desbloqueio no Android, então o *keystore* já estará inicializado. Caso contrário, o Android exigirá que o usuário cadastre um PIN ou padrão de desbloqueio, como mostra a figura 19.

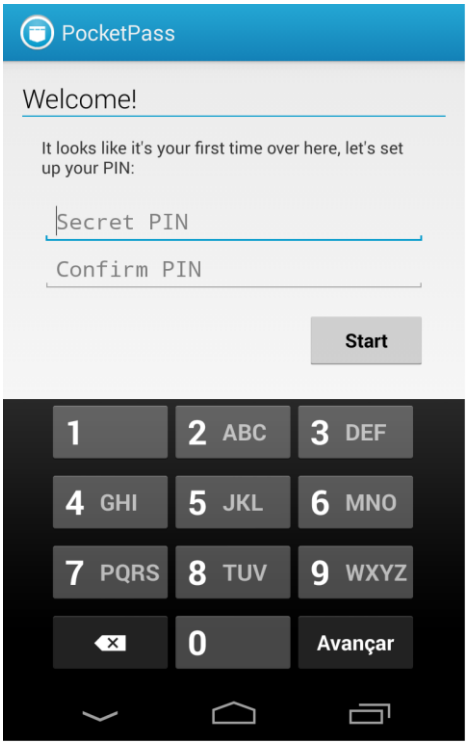
Figura 19 - Inicialização do *keystore* pelo Android



Fonte: elaborado pelo autor

Uma vez tendo inicializado o *keystore*, o usuário será solicitado a cadastrar um código de segurança no próprio aplicativo para proteger as identidades instaladas de acesso não autorizado. Esse código será utilizado nas derivações criptográficas para proteger as identidades além da cifra já realizada pelo *keystore* e também informado quando a aplicação for acessada após um longo período de inatividade. O PIN é persistido no *keystore* após um processo de *hashing* iterativo. Os algoritmos, iterações e detalhes são apresentados na seção de implementação.

Figura 20 – Tela de cadastro do PIN de usuário



Fonte: elaborado pelo autor

Por fim, o módulo provedor age em segundo plano criando uma identificação do dispositivo. São registrados informações de *hardware*, como *Serial Number* (SN), e gerado um identificador aleatório para a instalação. Esse identificador de instalação e o SN são enviados à organização em eventuais requisições de certificados.

O módulo verificador, por sua vez, não requer nenhuma inicialização prévia, carregando informações a partir da central apenas durante sua execução, a partir da chave organizacional informada no momento da solicitação da captura, como detalhado mais a frente.

4.3.3 *Instalação de identidade no provedor*

A instalação de identidade é o processo pelo qual um usuário realiza uma requisição de certificado a uma organização e recebe sua reposta, instalando então a identidade recebida no seu módulo provedor junto às chaves geradas pelo próprio módulo. A partir da instalação, a identidade pode ser usada livremente para identificação junto ao módulo verificador. O fluxo de comunicação que se segue é ilustrado na figura 21:

1. uma novo identidade é necessária;
2. o módulo provedor inicia a geração de par de chaves em segundo plano;
3. o módulo provedor solicita ao usuário suas credenciais organizacionais;
4. o módulo provedor apresenta uma tela de carregamento;
5. o módulo provedor contata a central, por conexão segura, com o e-mail informado para inferência da organização e recebimento do endereço e certificado HTTPS. Essa inferência de domínio vem de uma das informações negociadas quando uma organização se cadastra junto a central;
6. o módulo provedor inicia a construção de uma requisição de certificado no formato PKCS#10 e assina a requisição com o par gerado no passo 2. O formato PKCS#10 é um formato padronizado para solicitação de certificados com regulamentação no IETF, sendo suportado nativamente pelo provedor de criptografia BouncyCastle utilizado na solução;
7. mantendo secreta sua chave privada, o módulo provedor envia a requisição de certificado à organização junto às credencias informadas pelo usuário, por conexão segura;
8. o módulo organizacional recebe a requisição a inicia sua análise. Antes de gerar e assinar o certificado, a análise verifica:
    a) se o IP do usuário está banido devido à suspeita de ataque força bruta e deve ser imediatamente rejeitado;
    b) se a requisição contém dados válidos sobre o *smartphone* requisitante, como SN e identificador de instalação, rejeitando se ausente;

c) se as credenciais organizacionais são pertencentes e válidas, rejeitando no caso contrário e registrando-se o erro para possível detecção de ataque futura;

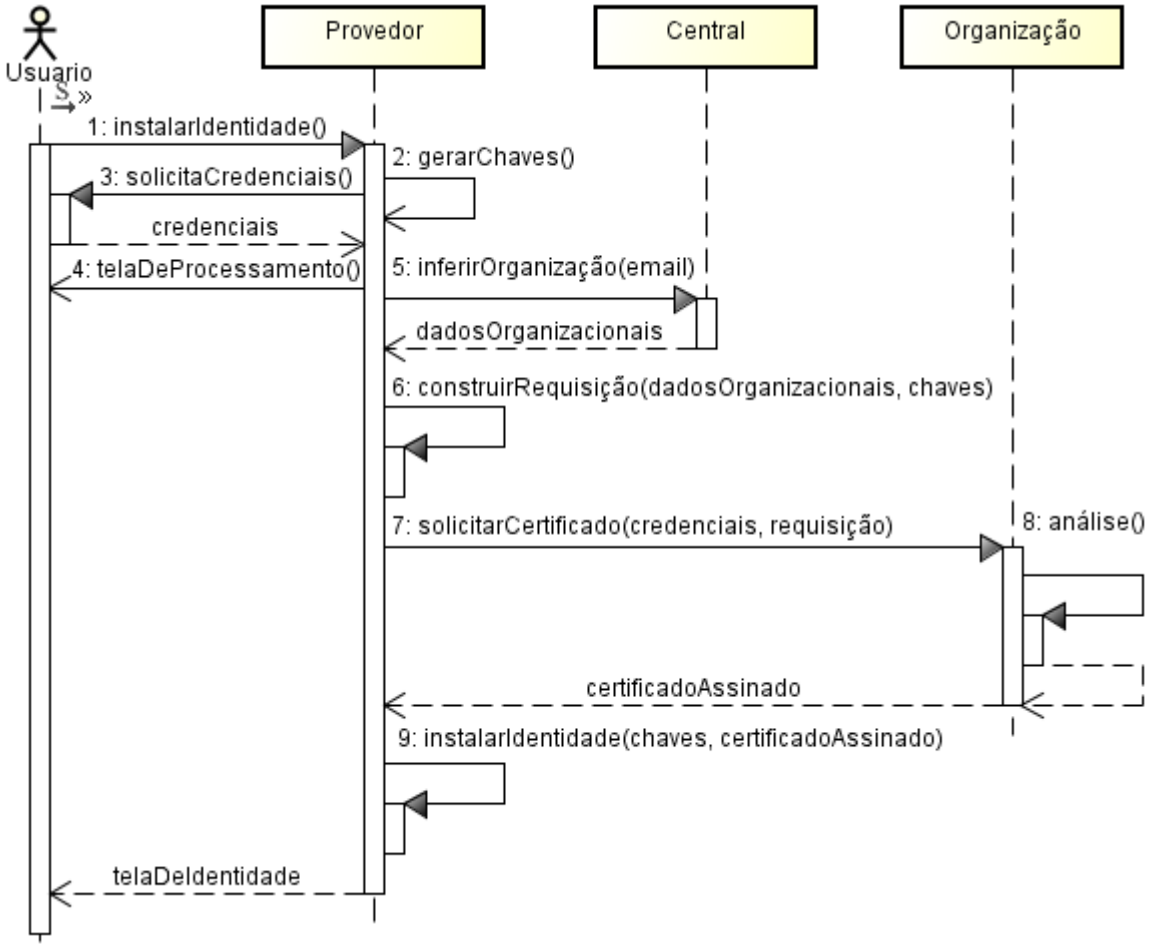
d) se a assinatura na requisição é válida com relação a chave pública informada na mesma, rejeitando se inválida;

e) se já existe um registro com a chave pública informada, rejeitando se existente;

f) se já existe um registro válido para as credenciais informadas. Em caso positivo, verifica se foi adicionada uma permissão excepcional devido à revogação do certificado anterior, rejeitando se não permitido;

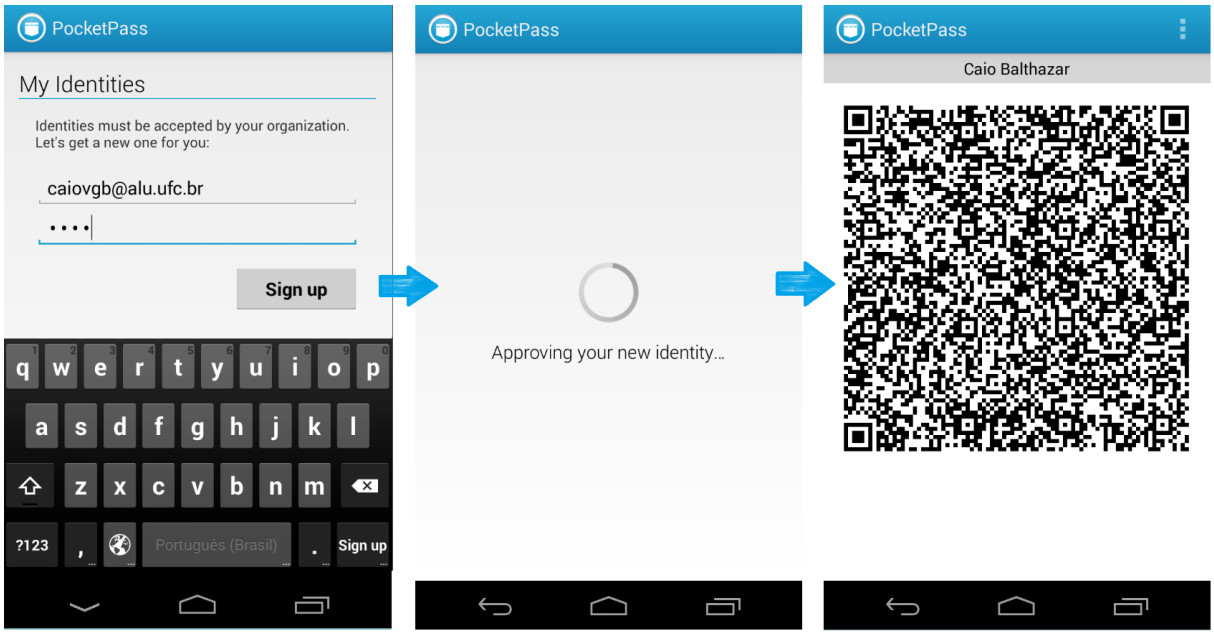
9. o módulo provedor recebe o certificado e o armazena, cifra a chave privada com *password based encryption* (PBE) iterativa baseada no PIN do usuário e armazena a cifra no *keystore* do sistema. Em seguida, apresenta a tela de identidade ao usuário onde serão gerados códigos QR.

Figura 21 - Diagrama de sequência da instalação de identidade



Fonte: elaborado pelo autor

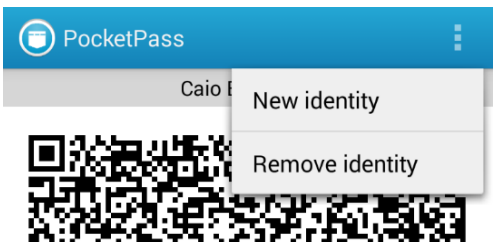
Figura 22 - Fluxo de telas da instalação de identidade



Fonte: elaborado pelo autor

O processo de instalação de identidade tem início automaticamente sempre que o módulo provedor móvel é iniciado sem identidades instaladas ou pela escolha manual do usuário. O usuário pode ainda remover identidades, com perda permanente das chaves privadas que a suportavam, ou ainda instalar uma identidade mais recente para substituir uma identidade expirada, cenário comum no modo de operação por certificados curtos. Ao instalar uma nova identidade com uma mesma credencial numa mesma organização, a credencial antiga será automática atualizada no módulo provedor, removendo a identidade expirada.

Figura 23 - Acesso às opções de gerenciamento de identidades no módulo provedor



Fonte: elaborado pelo autor

Embora o processo de instalação envolva bastante comunicação, em condições favoráveis ele pode ser completado em menos de 10 segundos, como mostra a seção sobre desempenho.

4.3.4 *Posicionamento organizacional do verificador*

Como mencionado no início da seção, o módulo verificador não requer inicialização, realizando seu posicionamento organizacional baseando-se na informação repassada ao módulo pela solicitação de captura, que vem de aplicativos externos. No entanto, no cenário que um usuário abra o módulo manualmente e consequentemente não existam dados organizacionais, ele será solicitado a indicar a chave organizacional na qual deseja operar.

No caso de reuso por outro aplicativo, a comunicação nativa da plataforma é utilizada para receber parâmetros e retornar os resultados. O reuso da solução se baseia simplesmente na solicitação de captura ao módulo verificador. Nenhuma integração de código é necessária. No Android, aplicativos podem fazer solicitações a outros aplicativos para obter algum tipo de resultado. Esse é o principal mecanismo de extensão implementado no aplicativo verificador. Essa requisição acontece por meio de uma *intent* explícita, objeto capaz de carregar também os parâmetros para execução de operações.

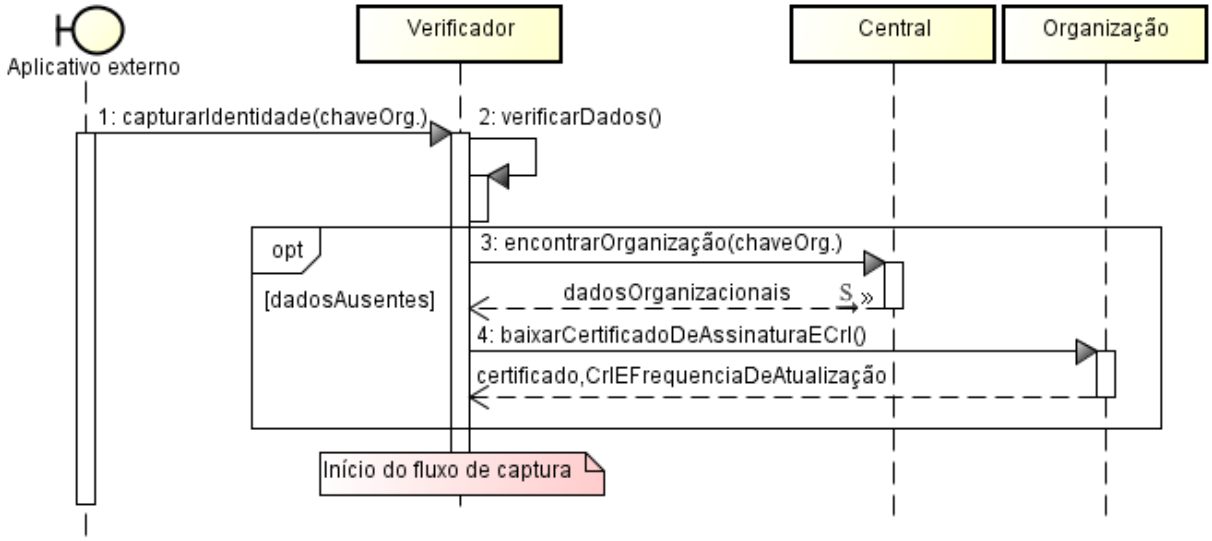
O módulo verificador é capaz de receber vários parâmetros, como chave organizacional, limite de capturas e necessidade de receber integralmente os dados capturados, sendo a chave organizacional o principal parâmetro. Assim, aplicativos que se integram ao módulo verificador podem disparar uma requisição de captura já direcionada à organização desejada, fazendo com que o módulo verificador inicie automaticamente na tela de captura, sem necessidade de intervenção do usuário.

O fluxo envolvido é detalhado na figura 24:

1. o aplicativo externo dispara uma *intent* contendo a chave organizacional e, opcionalmente, os demais parâmetros;
2. o módulo verificador checa os dados previamente armazenados e verifica se há a necessidade de solicitar alguma informação à central ou à organização.
3. no caso em que não há dados previamente armazenados é feita uma consulta à central, por conexão segura, para obter o endereço e certificado HTTPS da organização;
4. o módulo verificador então solicita à organização: seu certificado usado para derivar os certificados de usuários (de preferência diferente do certificado HTTPS), sua versão atual da CRL e sua frequência de atualização da mesma;

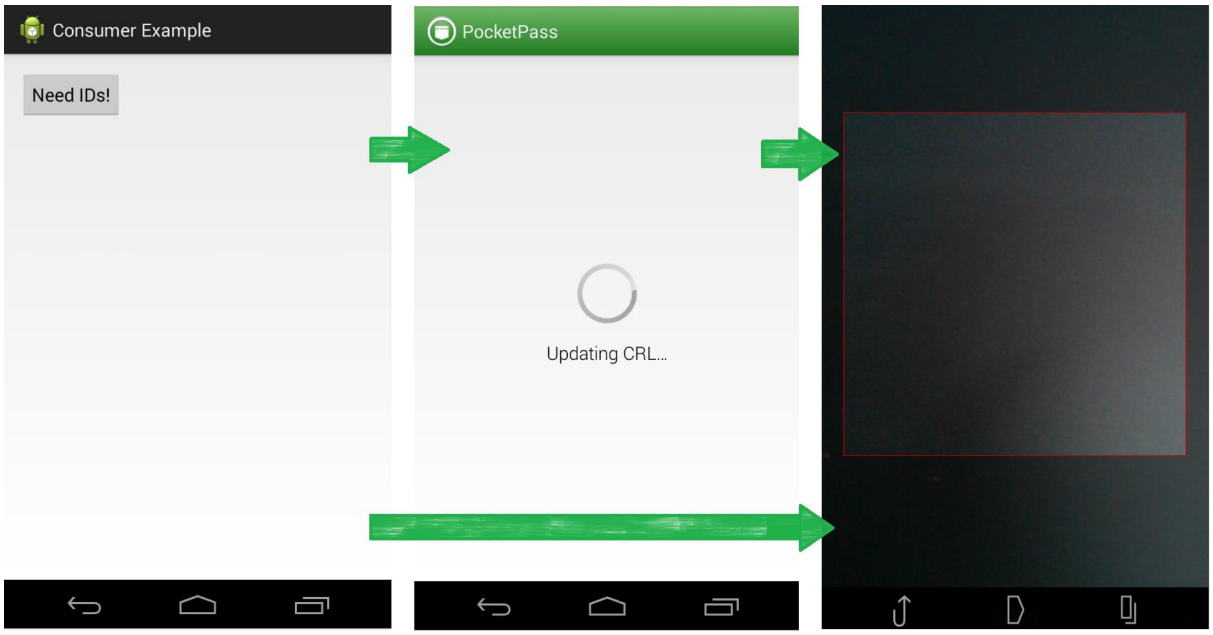
5. tem início o processo de captura e verificação, a partir do passo 2 se as informações já houvessem sido obtidas previamente ou partir do passo 4 se essa é primeira interação com essa organização.

Figura 24 - Diagrama de sequência do posicionamento organizacional automático



Fonte: elaborado pelo autor

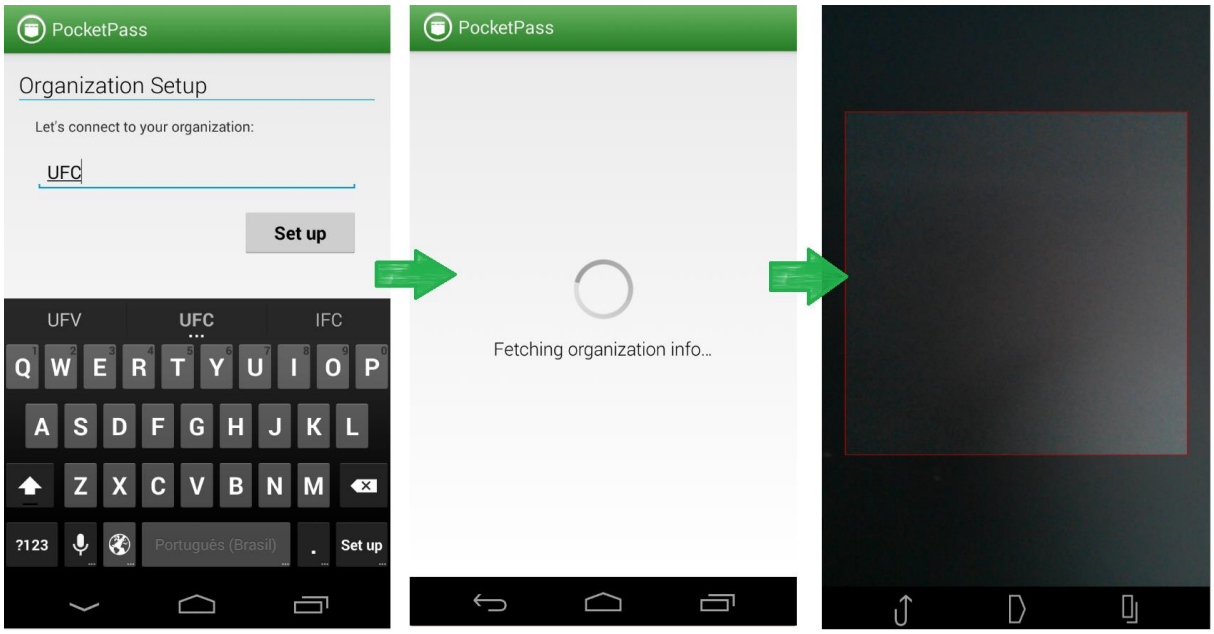
Figura 25 - Fluxo de telas do posicionamento organizacional do módulo verificador a patir de requisição de um aplicativo externo, com e sem a necessidade de atualização de dados



Fonte: elaborado pelo autor

No caso de primeiro uso direto pelo usuário, o processo segue um fluxo levemente diferente, onde o usuário informa a chave organizacional desejada, substituindo a função exercida pelo parâmetro externo no cenário anterior.

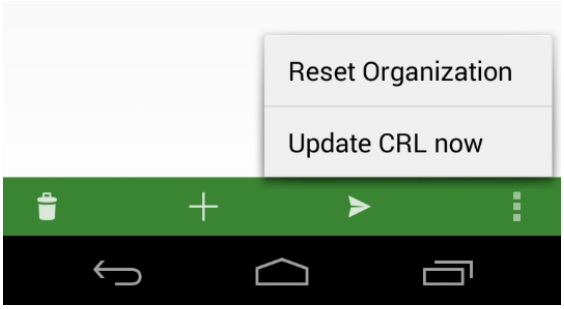
Figura 26 - Exemplo do fluxo de telas de uso direto do módulo verificador pelo usuário



Fonte: elaborado pelo autor

Caso o módulo contenha dados previamente instalados, ele não exigirá que o usuário reinforme a organização, abrindo diretamente a tela de captura. Não obstante, a opção de mudar a organização estará disponível ao usuário no menu da aplicação. Os dados armazenados no módulo verificador incluem apenas os certificados das organizações (o de comunicação e o de assinatura dos certificados dos usuários), seus endereços, suas CRLs e frequências de atualização. É importante lembrar que nenhum desses é confidencial.

Figura 27 - Acesso às opções de gerenciamento de dados no módulo verificador



Fonte: elaborado pelo autor

4.3.5 *Verificação de identidade*

O principal fluxo de interação da arquitetura acontece envolvendo os módulos provedor e verificador: a verificação de identidade. Esse fluxo integra todos os elementos da

solução: modelo de autenticação, sincronia temporal, criptografia, acesso aos dados sensíveis, geração dos códigos visuais, verificação dos códigos, controle de sessão e integração com aplicativos externos.

Começando pelo módulo provedor, desde que o fluxo não venha do processo de instalação como mostrado anteriormente, ele se inicia ao abrir a aplicação a partir da *home* do Android e confirmar o código PIN do usuário. Enquanto o usuário digita seu código, o gerenciador temporal age em segundo plano garantindo que a aplicação tenha informação de tempo necessária para iniciar a construção de códigos imediatamente tão logo o usuário libere o acesso à próxima tela.

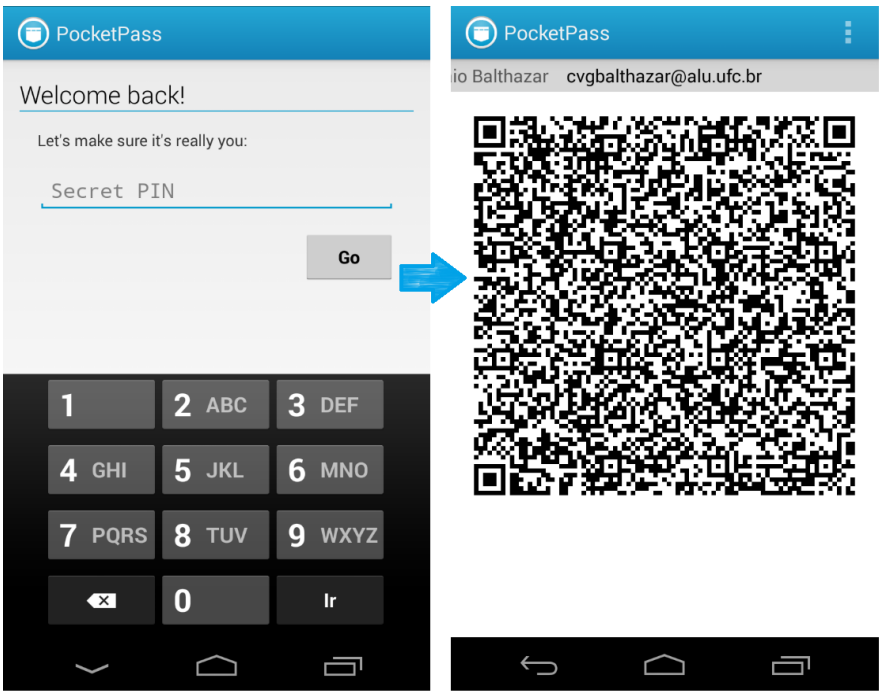
O código PIN protege a tela de identidade sempre que ela é acessada após terem se passado mais de três minutos desde a última digitação correta do PIN. No caso de tentativas incorretas, a aplicação dispõe de uma trava de segurança detalhada posteriormente.

Uma vez digitado o PIN, o usuário será levado à tela de identidade onde terá início a construção dos códigos QR. O módulo provedor irá exibir todas as identidades instaladas de maneira navegável ao arrastar lateralmente a tela. Os códigos QR se renovam automaticamente a cada 15 segundos. O processo de construção de um código QR envolve os seguintes passos:

1. recuperação de informação de tempo a ser assinada;
2. recuperação da chave privada da identidade a partir do *keystore* do sistema;
3. recuperação do certificado da identidade;
4. decifração da chave utilizando a senha PBE derivada do PIN;
5. assinatura de uma *Cryptographic Message Syntax* (CMS) encapsulando a informação de tempo, utilizando a chave privada e o certificado. A CMS é um formato de encapsulamento de dados para encriptação que contribui para a segurança da assinatura ao mesmo tempo que carrega informações do algoritmo e certificado utilizados, permitindo que suítes criptográficas possam verificar sua assinatura sem a utilização de código específico à aplicação.
6. compressão da CMS assinada para tornar ainda mais eficiente o reconhecimento do código QR ao diminuir sua densidade;

Uma vez comprimida, a CMS será então codificada em QR para exibição na tela do dispositivo. O módulo provedor calcula a maior dimensão de exibição do código QR almejando atingir a maior área possível, facilitando a leitura do mesmo.

Figura 28 - Fluxo de acesso a tela de identidade para autenticação



Fonte: elaborado pelo autor

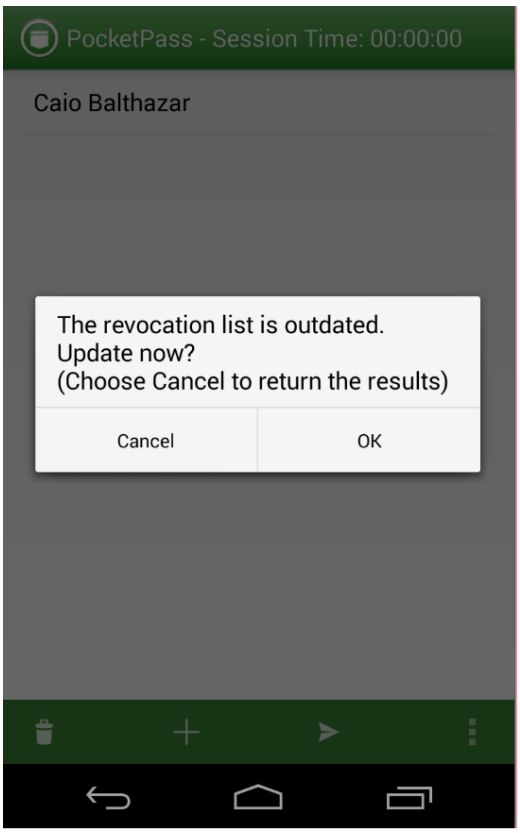
Neste momento, o dispositivo com o módulo provedor se encontra pronto para ser autenticado pelo módulo verificador.

É esperado que neste momento o módulo verificador já tenha iniciado a sessão de autenticação e já esteja em sua tela de captura para receber o código QR. O gerenciamento de sessão de autenticação é uma das operações mais complexas do módulo verificador. A sessão é mantida válida sempre que há uma CRL cobrindo o período de tempo atual com sua validade. A sessão é renovada automaticamente sempre que possível através da obtenção de novas CRLs, quando necessário, tudo em segundo plano. O gerenciamento de processos assíncronos em ambos os módulos móveis é uma das técnicas que permitem a agilidade do processo de autenticação.

No caso da impossibilidade de atualização automática de CRLs, devido a problemas de conectividade e houver a expiração da sessão de autenticação, o usuário será consultado para tomar uma decisão: conseguir conectividade e fazer uma atualização de CRL ou retornar apenas os resultados obtidos até então, antes da expiração, rejeitando qualquer

resultado recebido após o fim da validade. O princípio por traz desse mecanismo é que o módulo verificador, por segurança, apenas opera coberto pela CRL.

Figura 29 - Consulta ao usuário sobre sessão expirada



Fonte: elaborado pelo autor

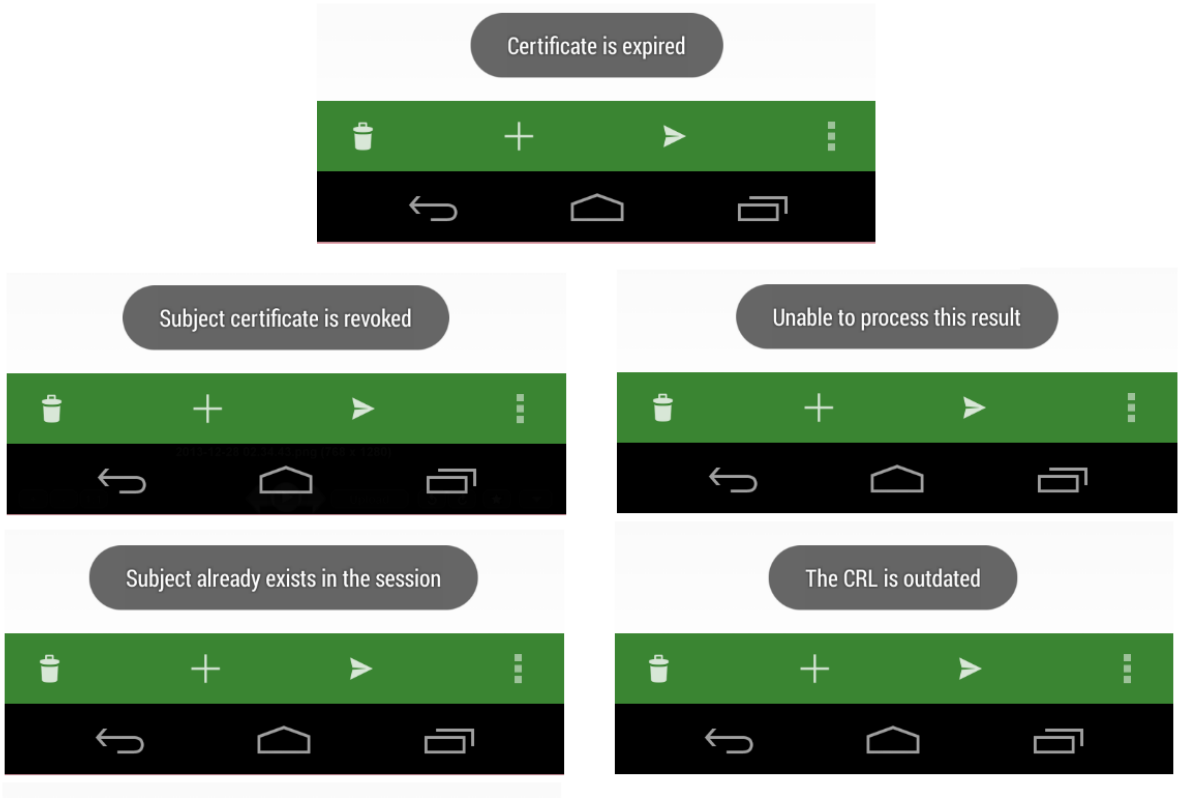
Independentemente do modo de inicialização do módulo verificador, seja por um aplicativo externo ou pelo usuário como mostrado anteriormente, a tela de captura será rapidamente exibida para leitura do primeiro código QR. Após a realização de uma captura, o código será interpretado e verificado, seguindo os passos:

1. descompressão da CMS;
2. extração do certificado do assinante;
3. verificação da assinatura com relação à chave pública no certificado;
4. verificação se o certificado foi assinado pela organização no contexto do módulo verificador;
5. verificação de validade (expiração) do certificado;
6. extração do dado assinado na CMS;

7. verificação da informação temporal assinada na CMS. A informação deve estar na janela especificada de 30 segundos com uma tolerância de 15 segundos para erros de sincronia, ou seja, deve ter no máximo 45 segundos de idade e pode estar adiantada até 15 segundos;
8. verificação se o sujeito identificado no certificado já está presente na sessão atual de autenticação;
9. verificação de revogação do certificado junto a CRL;
10. verificação se a sessão de autenticação ainda está válida ou está expirada;

Se qualquer um desses passos falhar, o resultado obtido será rejeitado e o agente verificador será alertado.

Figura 30 - Alguns exemplos dos possíveis erros de verificação detectados pelo módulo verificador (lista não exaustiva)

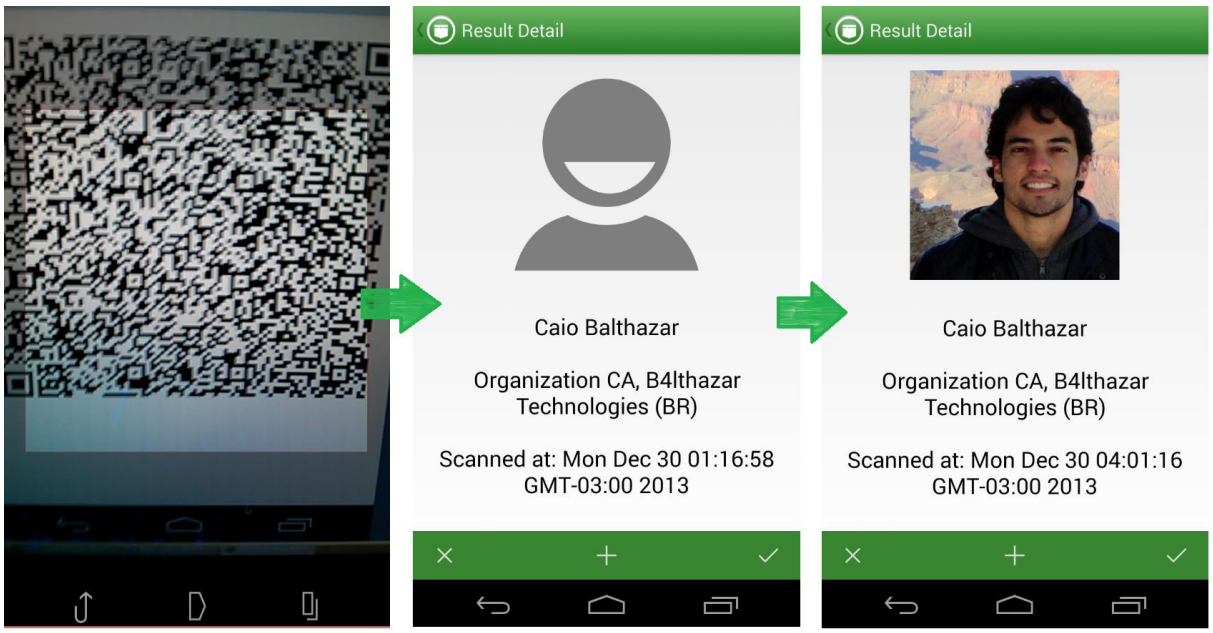


Fonte: elaborado pelo autor

No caso de uma verificação bem sucedida, o agente verificador será levado a uma tela de exibição e confirmação da identidade recebida. Nessa tela, o módulo verificador irá contatar o módulo organizacional para a obtenção de dados complementares sobre a identidade detectada. No entanto, os dados adicionais são meramente complementares e não é

necessário que seu recebimento seja bem sucedido. No caso do protótipo apresentado, caso haja conectividade, uma foto é baixada (também por HTTPS) e exibida para confirmação da identidade.

Figura 31 - Exemplo de verificação bem sucedida com recebimento de informações adicionais

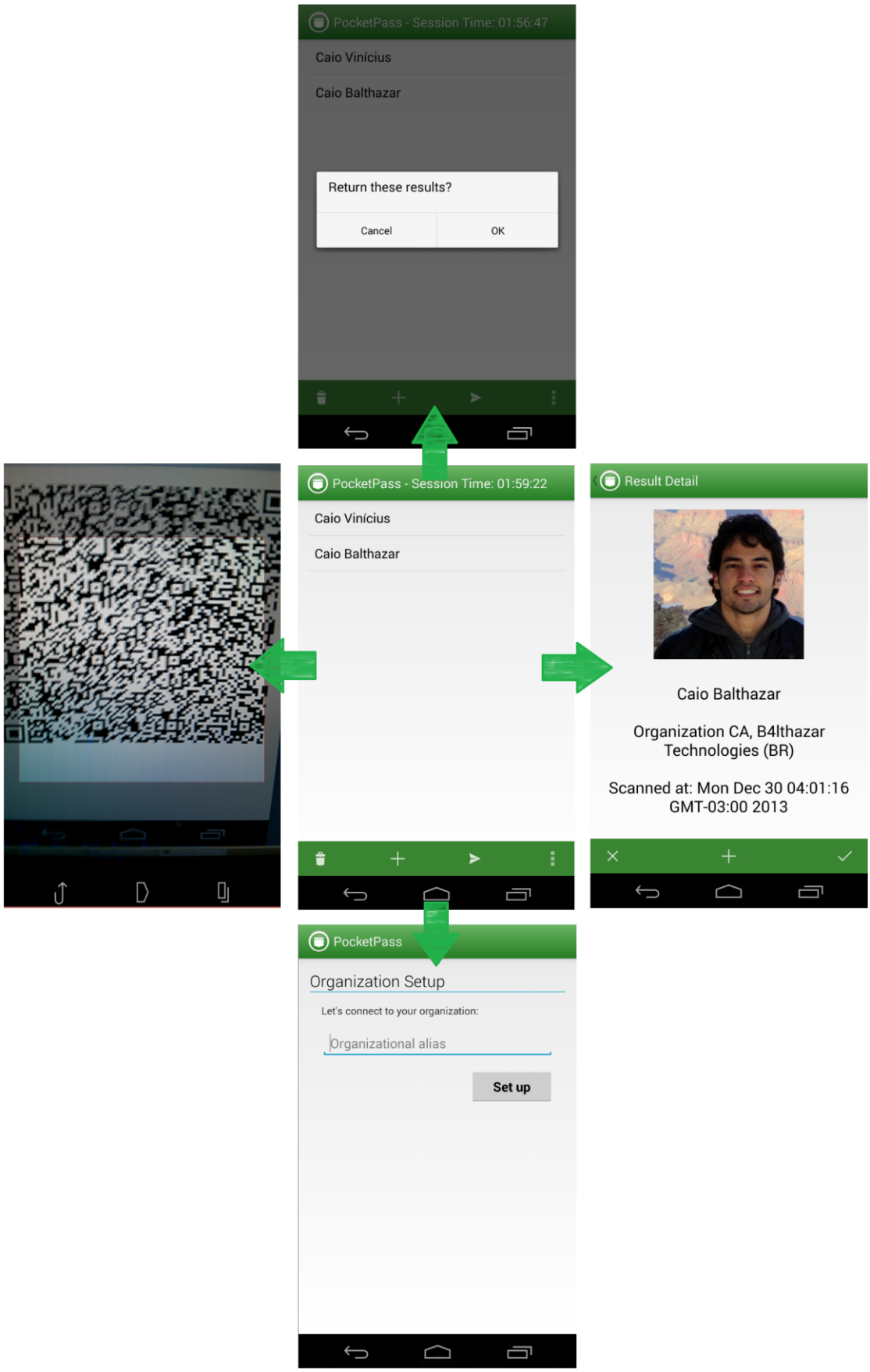


Fonte: elaborado pelo autor

Na tela de detalhe, o agente verificador poderá rejeitar o resultado, aceita-lo e já iniciar uma novo escaneamento ou apenas aceita-lo e retornar à lista de resultados. Em ambas as opções de aceitação, o módulo verificador irá checar se a aplicação externa que o iniciou informou algum parâmetro de limite de resultados e se esse limite já foi atingido. Caso o limite de resultados tenha sido especificado e atingido, o módulo verificador irá retornar os resultados ao aplicativo externo.

No caso de aceitação e retorno à lista, o agente verificador poderá navegar pelos resultados obtidos e executar remoções individuais, limpar todos os resultados, capturar novas identidades, retornar os resultados, trocar de contexto organizacional (perdendo os resultados atuais) ou atualizar a CRL manualmente.

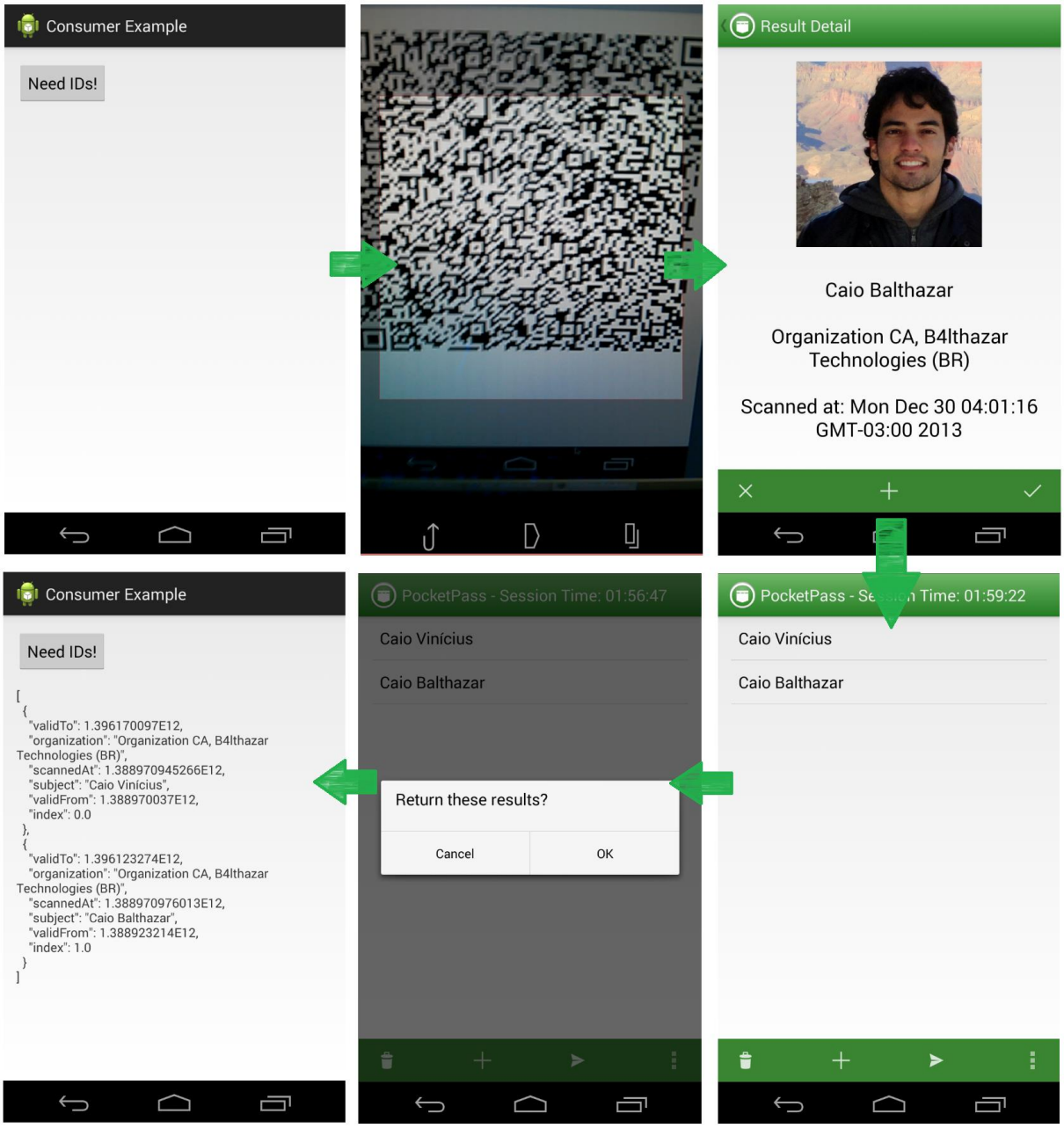
Figura 32 - Navegabilidade do módulo verificador



Fonte: elaborado pelo autor

Por fim, ao retornar os resultados, o aplicativo externo receberá as identidades solicitadas e poderá emprega-las em qualquer uso que lhe seja de interesse.

Figura 33 - Ilustração do recebimento de resultados pelo aplicativo externo consumidor



Fonte: elaborado pelo autor

Opcionalmente, o aplicativo externo poderá informar um parâmetro indicando que deseja receber também os dados contidos na CMS validada, como certificado do usuário e a assinatura da informação temporal.

4.3.6 *Revogação*

O procedimento de revogação é surpreendentemente simples através da arquitetura proposta. Uma vez detectado o comprometimento de seu dispositivo, o usuário poderá revogar seus próprios certificados rapidamente através de um serviço disponível no módulo organizacional disponível na URL '/certificate/revocate' sob o endereço do módulo. Como qualquer acesso HTTPS via *browser*, é importante que o usuário verifique o certificado organizacional antes de submeter suas credenciais.

Ao receber as credenciais do usuário e validá-las, o módulo organizacional irá buscar os certificados atualmente ativos sob aquelas credenciais e irá acrescentá-los à lista CRL. Assim como o serviço de solicitação de certificados, o módulo organizacional tenta detectar ataques de força bruta a esse serviço rejeitando imediatamente requisições de endereços que tenham feito repetidas tentativas mal sucedidas de validar as credenciais. Uma outra medida comum é uso de *captchas*, não implementado no protótipo.

Embora não tenha sido implementado no protótipo apresentado, um mecanismo de revogação interna por parte dos administradores organizacionais é perfeitamente viável.

4.4 Implementação da arquitetura

Parte do esforço de implementação da arquitetura se concentrou no levantamento do estado da arte criptográfica nos diversos contextos envolvidos pelo trabalho, principalmente para a tomada de decisões adequada. Durante esse levantamento foram detectadas algumas brechas de segurança em componentes de *software* relacionados ao trabalho, requerendo que essas brechas fossem contornadas. Algumas das mais significativas são:

4.4.1 *Android PRNG*

Em meados de 2013, uma falha no gerador de números pseudoaleatórios do Android foi descoberta graças ao seu envolvimento em um ataque à cadeia Bitcoin (STEVENSON, 2013), indicando que o gerador poderia ser previsível. No entanto, essa falha afeta não só aplicativos Bitcoin, mas todo tipo de *software* que utiliza o gerador de números

da plataforma para meios criptográficos, corrompendo a segurança de algoritmos de segurança.

Para consertar o problema, o time de desenvolvedores Android publicou um trecho de código que deveria ser inserido manualmente nas aplicações para contornar o defeito (KLYUBIN, 2013). Esse trecho de código foi integrado a ambos os módulos provedor e verificador.

4.4.2 *Android PBKDF2*

Ao final de 2012, foram encontrados indícios de que a função PBKDF2 no Android conteria falhas parciais em sua implementação, como mencionado no *Android Open Source Project* (KBSRIRAM, 2012). Devido a esse problema e ao fato do *keystore* do sistema utilizar essa função, as cifras baseadas em chaves (PBE) utilizadas no módulo provedor substituem essa função pelo uso da biblioteca Jasypt com seu algoritmo PBE baseado na implementação BouncyCastle dos algoritmos SHA-256 e AES-128 com ‘sal’ aleatório. A eficiência dessa implementação é tamanha que foi possível utilizar mais de 10 mil iterações PBE no módulo provedor sem impacto significativo na experiência do usuário.

4.4.3 *Apache HTTP client 3*

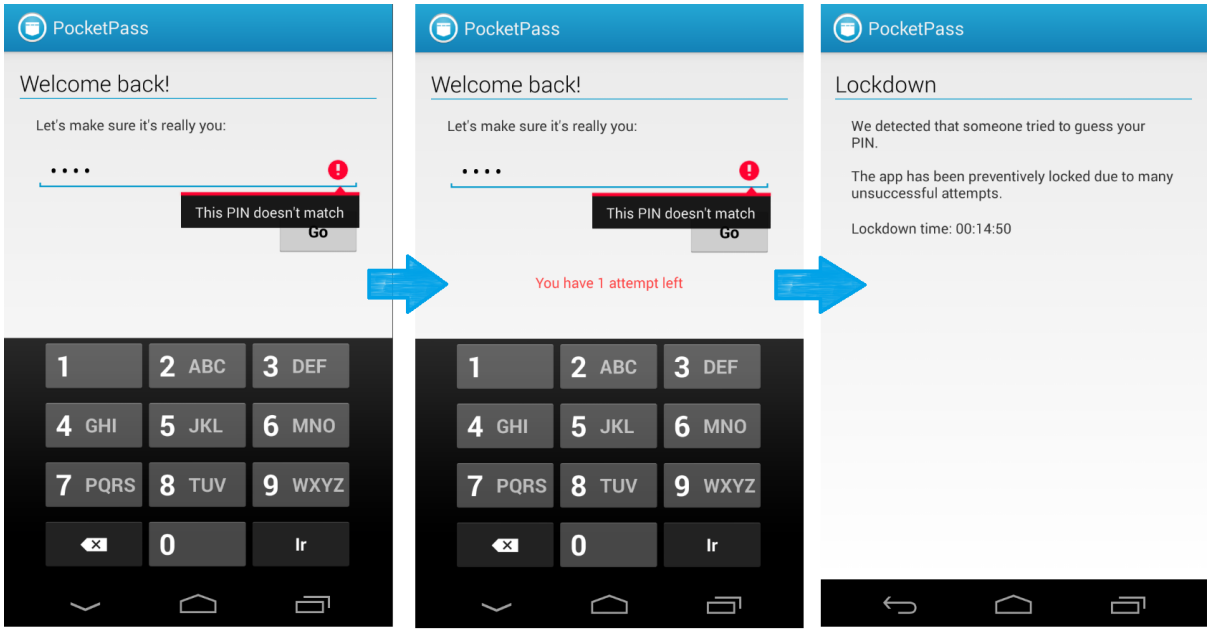
Assim como apontam Georgiev, Jana, e Boneh (2012) a implementação do cliente Apache HTTP versão 3, no qual se baseiam algumas versões do cliente HTTP do Android (e outros milhares de *softwares* em todo o mundo), contém falhas de segurança na verificação de certificados durante o uso do protocolo HTTPS. Oportunamente, os módulos provedor e verificador foram implementados sobre o cliente HTTP lançado pela empresa Square, conhecido com OkHttp, que tem ganhado bastante popularidade no último ano devido ao seu destaque na Google I/O 2013. Mesmo assim, testes foram realizados durante o desenvolvimento dos módulos móveis para assegurar que o cliente OkHttp de fato não cometia as mesmas falhas que o cliente Apache.

4.4.4 *Medidas de segurança extra*

Algumas medidas de segurança extras, não relacionadas a falhas, também foram incluídas na implementação. Como exemplos, tem-se:

a) bloqueio preventivo do aplicativo provedor por 15 minutos após 4 tentativas incorretas de PIN;

Figura 34 - Bloqueio preventivo do aplicativo provedor



Fonte: elaborado pelo autor

b) desabilitação de *screenshots* na tela de códigos QR, dificultando a ação de um usuário desonesto;

c) indicação explícita no *manifest* da aplicação indicando à plataforma Android que as funções aplicação não podem ser acessadas diretamente (exceto pela chamada de extensão do módulo verificador) ou sequer debugadas;

d) a já mencionada detecção de ataques força bruta aos serviços do módulo organizacional;

e) uso de certificados diferentes para encriptação de dados (SSL/TLS) e assinaturas (derivação de certificados);

f) respeito às recomendações NIST para algoritmos de *hash* (SHA-256) e tamanhos de chave criptográficas (ECC e RSA);

g) delegação das operações criptográficas da solução ao provedor BouncyCastle (ou SpongyCastle, no caso Android) através de JCE. Para operações de *hashing* e *password based encryption* PBE, se utilizou a biblioteca Jasypt.

4.4.5 *Armazenamento de segredos no Android*

Como apontado na revisão bibliográfica, apesar de ter boas qualidades de segurança, o *keystore* do sistema Android é burocraticamente restritivo a aplicações não-nativas antes da versão 4.3. No entanto, tendo a agilidade de autenticação como um dos principais objetivos, era inviável que o módulo provedor utilizasse o paradigma de interação imposto pelo sistema. Simultaneamente, restringir a aplicação apenas à parcela de usuários atualmente com Android 4.3 ou superior seria uma decisão de grande impacto na aplicabilidade da solução.

Nesse cenário, uma solução foi desenvolvida seguindo o trabalho de Elenkov (2012). Utilizando-se da abertura de código da plataforma e das contribuições disponíveis no *Android Open Source Project (AOSP)*, foi possível implementar uma interface de acesso direto ao *keystore* do sistema mesmo em versões inferiores à 4.3. Para tanto, foi utilizado uma adaptação do código AOSP que acessa as funções do *keystore* com o uso de reflexão Java e *sockets*.

Vale ressaltar que a segurança do *keystore* se mantém intacta e o único diferencial da solução é sua capacidade de interagir com o *keystore* sem a necessidade de chamar telas externas. As mesmas regras de acesso limitado à aplicação criadora das informações ainda são impostas pelo *keystore*, que apenas interage com o mundo externo através de *sockets*, ou seja, de forma indireta.

A solução inspirada por Elenkov permite que o módulo provedor persista dados no *keystore* nativo transparentemente, não raro dezenas de vezes por uso da aplicação, fazendo uso de suas qualidades inatas de segurança.

4.4.6 *Ferramentas utilizadas nos módulos móveis*

Para a implementação dos módulos móveis, foi criado um submódulo com funcionalidades e ferramentas comuns a ambos provedor e verificador. As ferramentas utilizadas nesse módulo comum incluem:

a) SpongyCastle (TYLEY, 2013):

   – provedor criptográfico baseado no BouncyCastle.

b) RoboSpice (NICOLAS, 2013):

   – *framework* para gerenciamento de tarefas e requisições assíncronas no Android.

c) Retrofit (SQUARE INC, 2013):

   – *framework* para tradução de interfaces Java em interfaces *Representational State Transfer* (REST).

d) Jackson (FASTERXML LLC, 2013):

   – interpretação e conversão de objetos em *JavaScript Object Notation* (JSON).

e) ZXing (GOOGLE INC, 2013):

   – manipulação de códigos QR no Android.

Uma ferramenta utilizada exclusivamente no módulo provedor:

a) Jasypt (JASYPT, 2013):

   – criptografia simplificada para Java.

Uma ferramenta utilizada exclusivamente no módulo verificador:

a) Picasso (SQUARE INC, 2013):

   – transferência de imagens via rede de maneira simplificada no Android.

4.4.7 *Ferramentas utilizadas nos módulos servidores*

Assim como nos módulos móveis, os módulos servidores se baseiam em um sub módulo comum. As ferramentas compartilhadas são:

a) BouncyCastle (LEGION OF THE BOUNCY CASTLE INC, 2013):

   – provedor de criptografia.

b) Dropwizard (HALE, 2013):

– *framework* servidor REST de alto desempenho orientado a APIs do tipo serviço.

Uma ferramentas utilizada exclusivamente no módulo organizacional:

a) Jasypt (JASYPT, 2013);

4.4.8 *Interface do módulo organizacional*

Tabela 1 – Interface do módulo organizacional

Método HTTPS	**Caminho**	**Função**
GET	/crl	Fornece a CRL atual da organização
GET	/certificates	Fornece os certificados utilizados na assinatura da CRL e nos certificados dos usuários (derivação)
GET	/time	Fornece o tempo atual da organização
POST	/certificate/request	Recebe uma solicitação de certificado para análise
POST	/certificate/revocate	Recebe uma solicitação de revogação para análise

Fonte: elaborado pelo autor

4.4.9 *Interface do módulo central*

Tabela 2 – Interface do módulo central

Método HTTPS	**Caminho**	**Função**
GET	/organization	Fornece informações para o direcionamento organizacional, como certificado, endereço, e chave. Pode ser utilizado tanto com o parâmetro ‘inf’ correspondente a uma inferência baseada em uma credencial (e-mail) como com o parâmetro ‘alias’, correspondente a uma chave

organizacional

Fonte: elaborado pelo autor

5 APLICAÇÃO E ESTUDO DE DESEMPENHO

5.1 Desempenho: RSA e ECC

Assim como mencionado na revisão bibliográfica, ECC tradicionalmente atinge tempos de resposta menores que RSA. No entanto, o tamanho de chave utilizado para RSA, seguindo a recomendação do NIST, foi de 2048 bits. Esse valor foi escolhido não apenas por ser recomendado, mas também visando minimizar o tamanho da assinatura gerada, de modo a possibilitar sua transferência via códigos QR.

Enquanto o tamanho de chave recomendado pelo NIST para ECC na mesma época seja 224 bits, equivalente a 2048 bits RSA, navegadores modernos não implementam suporte à curva ECC com esse tamanho, dificultando significativamente o processo de desenvolvimento e testes do protótipo. Com isso, o valor escolhido para o tamanho de chaves ECC foi o mínimo implementado pelos navegadores modernos, que está um nível acima da recomendação atual: 256 bits. Esse nível de proteção equivale a uma chave RSA de 3072 bits. A comparação foi, portanto, desnivelada beneficiando o RSA, podendo não apresentar a mesma vantagem para ECC como na literatura.

Dessa forma, para comparar a viabilidade das duas soluções, testes foram realizados com um dispositivo LG Nexus 4 E960 e um dispositivo Samsung Google Nexus 10 P8110 para comparar a performance da solução sob as duas alternativas de algoritmos assimétricos.

Os módulos organizacional e central foram implantados em um mesmo servidor e instalados com certificados assinados utilizando os respectivos algoritmos e tamanhos. Os servidores e os dispositivos móveis foram colocados sob a mesma rede *WiFi* local de modo a minimizar variações no desempenho de rede.

Os testes foram divididos em 4 métricas: tempo de instalação, tempo de geração de código, tempo de reconhecimento e tamanho da CMS gerada. Para cada tipo de teste, foram cronometradas 33 amostras automatizadas.

Para a medição do tempo de reconhecimento, o dispositivo Nexus 4 (com a menor tela) foi afixado a um suporte e o dispositivo Nexus 10 foi afixado a outro suporte de modo que sua câmera estivesse perfeitamente alinhada com a tela pequena do Nexus 4. O aparelho

menor recebeu a instalação do módulo provedor, gerando automaticamente códigos válidos usando os respectivos algoritmos, enquanto o dispositivo Nexus 10 executava, cronometradamente, o reconhecimento após cada código gerado. O processo de leitura foi automatizado para que não fosse necessário tocar nos aparelhos durante os testes. A iluminação também foi estabilizada através de um ambiente fechado com iluminação artificial.

5.1.1 *Tempo de instalação*

O tempo de instalação medido foi o tempo desde o início da criação do par de chaves até abertura da tela de identidade para geração do primeiro código QR, incluindo todo o processo de geração de chaves, construção da requisição de certificado, comunicações, análise, assinatura do certificado, PBE e persistência no *keystore*.

5.1.2 *Tempo de geração de código*

O tempo de geração de código corresponde ao tempo entre a abertura da tela de identidade e a exibição do código QR para autenticação, incluindo, portanto, o processo de criação, assinatura e compressão da CMS.

5.1.3 *Tempo de reconhecimento*

O tempo de reconhecimento medido foi o tempo desde a abertura da tela de captura até o fim da decodificação do código, incluindo, portanto, o processo de leitura QR, descompressão e verificação da CMS.

5.1.4 *Tamanho da CMS gerada*

Os tamanhos de CMS medidos foram os tamanhos das CMS geradas em bytes, após a compressão. Essa é a informação base para a codificação do QR.

5.1.5 *Resultados*

Os dados obtidos nos testes foram os seguintes:

Tabela 3 – Comparação de desempenho entre ECC 256 e RSA 2048 através dos testes automatizados

Medida	EC P-256 / SHA256withECDSA	RSA 2048 / SHA256withRSA
Média do tempo de instalação	**7,44 segundos**	**13,60 segundos**
Mínimo tempo de instalação	6,50 segundos	7,20 segundos
Máximo tempo de instalação	9,52 segundos	26,56 segundos
Desvio padrão do tempo de instalação	0,75 segundos	4,06 segundos
Média do tempo de geração de código	**1,39 segundos**	**1,07 segundos**
Mínimo tempo de geração de código	1,14 segundos	0,91 segundos
Máximo tempo de geração de código	1,64 segundos	1,47 segundos
Desvio padrão tempo de geração de código	0,13 segundos	0,10 segundos
Média do tamanho da CMS	**621.35 bytes**	**1225.04 bytes**
Mínimo tamanho da CMS	619 bytes	1221 bytes
Máximo tamanho da CMS	623 bytes	1228 bytes
Desvio padrão do tamanho da CMS	1,12 bytes	2,01 bytes
Média do tempo de reconhecimento	**4,13 segundos**	**8,28 segundos**
Mínimo tempo de reconhecimento	3,73 segundos	3,24 segundos
Máximo tempo de reconhecimento	5,28 segundos	68,05 segundos
Desvio padrão do tempo de reconhecimento	0,38 segundos	11,72 segundos

Fonte: elaborado pelo autor

Analisando os dados podemos chegar a algumas conclusões:

a) mesmo com um grau de segurança acima do RSA, ECC ainda apresenta melhor desempenho para a solução;

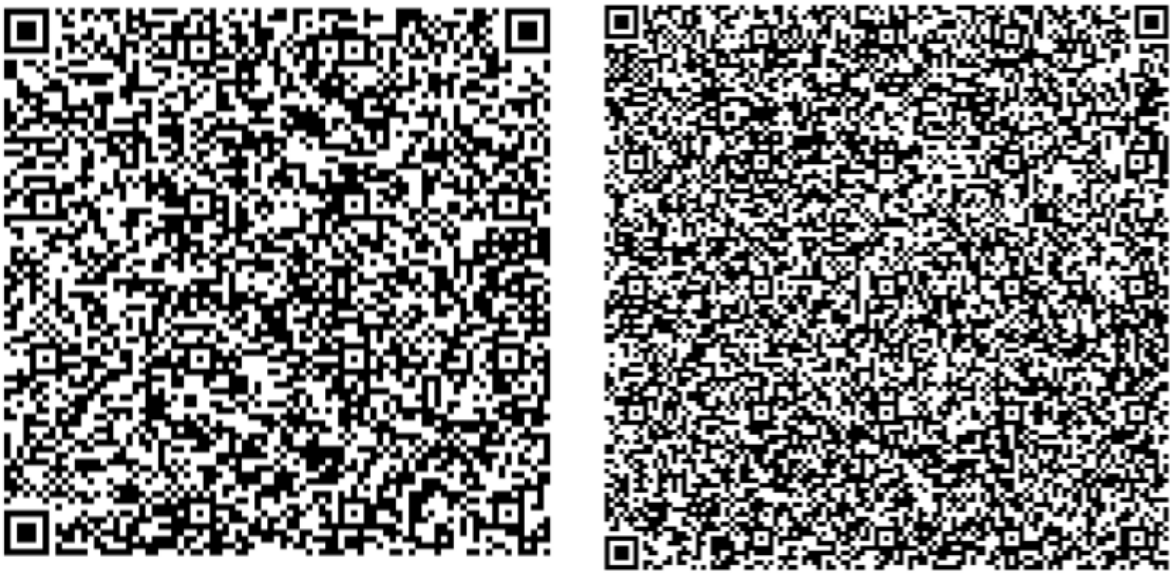
b) ECC apresenta melhores resultados em todas as métricas, exceto no tempo de geração de código onde o RSA é cerca de 0,3 segundos mais rápido;

c) após a compressão, a CMS RSA tem aproximadamente o dobro do tamanho da CMS ECC;

d) os resultados com ECC tendem a ter menor desvio padrão, sendo mais previsíveis, exceto novamente pela geração de código;

e) devido ao maior tamanho da CMS e consequentemente maior densidade no código QR, o tempo de reconhecimento RSA varia drasticamente, com a biblioteca de captura fazendo esforço consideravelmente maior, frequentemente reajustando do foco da câmera tentando aumentar a precisão da leitura do código;

Figura 35 - Comparação lado a lado das densidades de código QR entre ECC e RSA



Fonte: elaborado pelo autor

f) utilizando ECC, a solução implementada é capaz de seguir as recomendações de segurança NIST atuais e:

- instalar completamente uma nova identidade dentro de 10 segundos;
- gerar novos códigos QR autenticados dentro de 2 segundos;
- em condições ideais, interpretar e validar um código autenticado dentro de 6 segundos;

Com os resultados acima, fica claro o benefício que ECC traz à solução implementada, completando o conjunto de micro soluções integradas que compõem a

arquitetura aqui implementada. Sua prevalência performática sobre RSA mesmo em um nível de segurança acima chega ao ponto de ser considerada o fator viabilizador da solução. O protótipo implementado com essa tecnologia representa uma prova de conceito para a solução do problema de identificação organizacional móvel.

5.2 Comparação com trabalhos relacionados

Diversos autores abordaram desafios de autenticação móvel com diferentes propósitos. No entanto, o trabalho aqui desenvolvido se destaca das propostas anteriores por sua orientação explícita ao cenário organizacional e por sua abordagem simplificada de arquitetura ICP.

Autores como Hallsteinsen, Jorstad e Thanh (2007), McCune *et al* (2005), e Hassinen e Hypponen (2005) utilizam em seus modelos canais de comunicação como *Bluetooth*, SMS, redes de telefonia e *Internet* durante o processo de autenticação. Embora seus modelos sejam aplicáveis, aqui se propõe que o uso desses canais impõe limitações de conectividade, complexidade e por vezes inconveniência ao processo de autenticação. O modelo proposto aqui utiliza principalmente códigos de reconhecimento visual durante o processo de autenticação.

Autores como Tanvi e Sonal (2011), Bhutta e Ghafoor (2012), Liao (2009) e Cheng (2011) escolheram trabalhar com tokens OTP ou criptografia simétrica em seus modelos. No entanto, aqui se propõe que uma infraestrutura de chave pública oferece qualidades benéficas ao cenário de identificação organizacional, como a independência dos dados entre os dispositivos provedores e verificadores, a diminuição dos riscos relacionados à negociação e armazenamento de segredos compartilhados em dispositivos móveis, a capacidade inata de assinatura e não-repúdio, uma maior independência com relação à *Internet* após a instalação e possibilidade de extensão para funcionalidades futuras. Adicionalmente, a solução desenvolvida utiliza *smartphones*, ferramentas e técnicas criptográficas modernas para atacar a limitação computacional.

US Department of Defense (2006), Alazomai e Josang (2010) e Cheng (2011) baseiam seus modelos no uso de *hardware* ou plataformas especiais. Embora essa abordagem traga consigo uma segurança reforçada, essas necessidades não raro representam um fator inviabilizador para implantação da arquitetura. A arquitetura proposta aqui se baseia apenas em servidores e *smartphones* modernos comuns.

Abe, Itoh e Takahashi (2007) e Park, Lim e Park (2008) constroem uma arquitetura mista envolvendo diversas entidades e papéis comunicativos, como operadoras de telefonia, governos ou servidores intermediários para delegação de processamento. A visão deste trabalho é a de que simplicidade é um valioso elemento de autonomia e segurança. Ao envolver essas outras entidades ou papéis, a arquitetura está acrescentando pontos de manutenção e negociação. Aqui se propõe que a infraestrutura de chave pública em sua forma mais tradicional já é suficiente e adequada ao processo de identificação organizacional, sendo mais facilmente gerenciada se seus papéis forem autônomos, bem conhecidos e menos complexos.

Autores como McCune *et al* (2005), Busold *et al* (2013) e Chen, Wang e Wu (2013) orientam seus trabalhos a outros cenários de identificação como autenticação *ad-hoc*, *peer-to-peer*, ou ligadas domínios específicos. Em geral, não são relacionados à identificação organizacional, exigindo significativas adaptações em seus modelos para tornar suas abordagens soluções viáveis ao problema que esse trabalho se propõe a solucionar.

O problema de identificação organizacional móvel tem demandas práticas tão exigentes entre custos, segurança, usabilidade, manutenção e gerenciamento que muito dificilmente uma solução que não seja especificamente calibrada para esse cenário conseguirá aceitação significativa.

6 CONSIDERAÇÕES FINAIS

O objetivo geral do trabalho foi desenvolver uma solução para o problema da identificação organizacional móvel através de *smartphones*. Nesse sentido, foram atacados diversos subproblemas críticos para a sustentação da solução final.

Na base de tudo há uma arquitetura baseada apenas em *smartphones* e servidores, protegida por uma infraestrutura de segurança sob os padrões criptográficos atuais. Operando sobre ela há um modelo de autenticação com dois fatores, resistente e extensível. E finalmente, um armazenamento robusto de credencias através de uma implementação integrada à plataforma Android. Um algoritmo de sincronia temporal adaptativo permite que a solução se destaque operando em ambientes mesmo sem conectividade com a *Internet*.

O protótipo teve seu desempenho medido comparando-se tempos de resposta para ECC e RSA, com a implementação sob ECC obtendo tempos de resposta surpreendente menores, atestando seu enorme potencial no contexto móvel e embarcado.

A solução atendeu plenamente às expectativas iniciais e ainda foi adiante com sua arquitetura distribuída demonstrando flexibilidade dentro de múltiplas opções de operação e implantação. Sua extensibilidade permite reuso por virtualmente qualquer outra aplicação organizacional, com a construção de pequenos aplicativos comunicantes com o módulo verificador.

A agilidade da arquitetura se confirmou com a autenticação sendo realizada dentro de poucos segundos, utilizando apenas a base de dados organizacional existente e um canal de comunicação diferenciado. Sobretudo, atinge o equilíbrio buscado entre simplicidade e robustez, permitindo a autenticação de dois fatores a um relativo baixo custo.

Os avanços tecnológicos de *hardware* dos *smartphones* foram aliados às contribuições de *software* para o contorno das tradicionais limitações móveis. Foi através da interligação de pequenas soluções contribuídas individualmente que se alcançou o objetivo geral: uma solução extensível para autenticação organizacional móvel, performática e com dois fatores de segurança, um problema até então aberto na ausência de *hardware* especializado.

A solução tem um futuro extremamente promissor, com possibilidades de extensão para utilizar outros meios de comunicação como NFC, outras funcionalidades

possibilitadas por sua ICP, aplicações comerciais de vários tipos e integração com outros protocolos organizacionais.

REFERÊNCIAS

ABE, Tsuyoshi; ITOH, Hiroki; TAKAHASHI, Kenji. **Implementing identity provider on mobile phone**. In: ACM WORKSHOP ON DIGITAL IDENTITY MANAGEMENT, 2007, Fairfax. Fairfax: ACM, 2007. p.46-52.

ALZOMAI, Mohammed; JØSANG, Audun. **The mobile phone as a multi otp device using trusted computing**. In: NETWORK AND SYSTEM SECURITY, 4., 2010, Melbourne. Melbourne: IEEE Computer Society, 2010. p.75-82.

BARBOSA, Julio Cesar. **Criptografia de chave pública baseada em curvas elípticas**. S. 2003. 28 f. Monografia (Mestrado) - Curso de Redes, COPPE, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2003.

BHUTTA, Paisal Karim; GHAFOOR, Abdul; SULTAN, Shizra. **Smart phone based authentication and authorization protocol for spacs**. In: HIGH CAPACITY OPTICAL NETWORKS AND ENABLING TECHNOLOGIES, 9., 2012, Istanbul. Istanbul: IEEE Computer Society, 2012. p.127-131.

LEGION OF THE BOUNCY CASTLE INC. **The Legion of the Bouncy Castle**. 2013. Disponível em: <http://www.bouncycastle.org/specifications.html>. Acesso em: 06 jan. 2014.

BUSOLD, Christoph. et al. **Smart keys for cyber-cars: secure smartphone-based nfc-enabled car immobilizer**. In: ACM CONFERENCE ON DATA AND APPLICATION SECURITY AND PRIVACY, 3., 2013, San Antonio. San Antonio: ACM, 2013. p.233-242.

CANALYS. **Google's android becomes the world's leading smart phone platform**: Canalys reveals smart phone market exceeded 100 million units in Q4 2010. Palo Alto, 2011. Disponível em: <http://www.canalys.com/newsroom/google%E2%80%99s-android-becomes-world%E2%80%99s-leading-smart-phone-platform>. Acesso em: 06 jul. 2013.

CHEN, C.; WANG, K. e WU, T. A Scalable Transitive Human-Verifiable Authentication Protocol for Mobile Devices. **… Forensics and Security, …**, v. 8, n. 8, p. 1318–1330, 2013.

CHENG, Fred. **Security attack safe mobile and cloud-based one-time password tokens using rubbing encryption algorithm**. MOBILE NETWORKS AND APPLICATIONS. Hingham, v. 16, n. 3, p.304-336. 01 jun. 2011.

CHOI, Kyeongwon. et al. **A mobile based anti-phishing authentication scheme using qr code**. In: MOBILE IT CONVERGENCE, 2011, Gyeongsangbuk-do. Gyeongsangbuk-do: IEEE Computer Society, 2011. p.109-113.

DEFIGUEIREDO, Dimitri. **The case for mobile two-factor authentication**. SECURITY & PRIVACY, IEEE Computer Society. [S.l.], v. 9, n. 5, p.81-85. 26 set. 2011.

DENSO ADC. **QR code essentials**. 2011. Disponível em: <http://www.nacs.org/LinkClick.aspx?fileticket=D1FpVAvvJuo%3D&tabid=1426&mid=4802>. Acesso em: 06 jul. 2013.

ECE/CIS. **How NTP Works**. 2012. Disponível em: <http://www.eecis.udel.edu/~mills/ntp/html/warp.html>. Acesso em: 26 dez. 2013.

ELENKOV, Nikolay. **Using the ICS KeyChain API**. 2011. Disponível em: <http://nelenkov.blogspot.jp/2011/11/using-ics-keychain-api.html>. Acesso em: 26 dez. 2013.

ELENKOV, Nikolay. **ICS Credential Storage Implementation**. 2012. Disponível em: <http://nelenkov.blogspot.jp/2011/11/ics-credential-storage-implementation.html>. Acesso em: 26 dez. 2013.

FASTERXML LLC. **Jackson JSON Processor**. 2013. Disponível em: <http://wiki.fasterxml.com/JacksonHome>. Acesso em: 06 jan. 2014.

FOG CREEK SOFTWARE INC. **Trello**. 2013. Disponível em: <https://trello.com>. Acesso em: 06 jan. 2014.

GEORGIEV, M.; IYENGAR, S. e JANA, S. **The most dangerous code in the world**: validating SSL certificates in non-browser software. Proceedings of the …, 2012.

GOOGLE INC. **Google authenticator**. 2013. Disponível em: <http://https://code.google.com/p/google-authenticator/>. Acesso em: 04 jun. 2013.

GOOGLE INC. **Android, the world's most popular mobile platform**. 2013. Disponível em: <http://developer.android.com/about/index.html>. Acesso em: 06 jul. 2013.

GOOGLE INC. **Dashboards** | Android Developers. 2013. Disponível em: <http://developer.android.com/about/dashboards/index.html>. Acesso em: 26 dez. 2013.

GOOGLE INC. **ZXing**. 2013. Disponível em: <https://code.google.com/p/zxing/>. Acesso em: 06 jan. 2014.

GUTMANN, Peter; **Engineering Security**. 2013. Disponível em: < http://www.cs.auckland.ac.nz/~pgut001/pubs/book.pdf>. Acesso em 06 ago. 2013.

HALE, Coda. **Dropwizard**: Production-ready, out of the box. 2013. Disponível em: <http://dropwizard.codahale.com/>. Acesso em: 06 jan. 2014.

HALLSTEINSEN, Steffen; JORSTAD, Ivar; THANH, Do Van. **Using the mobile phone as a security token for unified authentication**. In: SYSTEMS AND NETWORKS COMMUNICATIONS, 2., 2007, Cap Esterel. Cap Esterel: IEEE Computer Society, 2007. p.68-74.

HASSINEN, Marko; HYPPONEN, Konstantin. **Strong mobile authentication**. In: WIRELESS COMMUNICATION SYSTEMS, 2., 2005, Siena. Siena: IEEE Computer Society, 2005. p.96-100.

IETF. **RFC 2986 -** PKCS #10: Certification Request Syntax Specification Version 1.7. 2013. Disponível em: < http://tools.ietf.org/html/rfc2986>. Acesso em: 26 dez. 2013.

IETF. **RFC 5246** - The Transport Layer Security (TLS) Protocol Version 1.2. 2013. Disponível em: < http://tools.ietf.org/html/rfc5246>. Acesso em: 26 dez. 2013.

IETF. **RFC 5652** - Cryptographic Message Syntax (CMS). 2013. Disponível em: <http://tools.ietf.org/html/rfc5652#section-1>. Acesso em: 26 dez. 2013.

JASYPT. **Jasypt**: Java simplified encryption. 2013. Disponível em: <http://www.jasypt.org/cli.html>. Acesso em: 06 jan. 2014.

KBSRIRAM. **Issue 40578**: Interoperability issue in PBKDF2 implementation. 2012. Disponível em: <https://code.google.com/p/android/issues/detail?id=40578>. Acesso em: 26 dez. 2013.

KEYLENGTH. **Keylength** - NIST Report on Cryptographic Key Length and Cryptoperiod (2012). 2013. Disponível em: <http://www.keylength.com/en/4/>. Acesso em: 26 dez. 2013.

KLYUBIN, Alex. **Some SecureRandom Thoughts**. 2013. Disponível em: <http://android-developers.blogspot.jp/2013/08/some-securerandom-thoughts.html>. Acesso em: 26 dez. 2013.

LAUREANO, Marcos Aurelio Pchek. **Gestão de Segurança da Informação**. 2005. Disponível em: <http://www.mlaureano.org/aulas_material/gst/apostila_versao_20.pdf>. Acesso em: 06 jul. 2013.

LEE, Young Sil. et al. **Online banking authentication system using mobile-otp with qr-code**. In: COMPUTER SCIENCES AND CONVERGENCE INFORMATION TECHNOLOGY, 5., 2010, Seoul. Seoul: IEEE Computer Society, 2010. p.644-648.

LIAO, Kuan-chieh. et al. **A one-time password scheme with qr-code based on mobile phone**. In: INC, IMS AND IDC, 5., 2009, Seoul. Seoul: IEEE Computer Society, 2009. p.2069-2071.

MAGALHÃES, D. K. S.; QUEIROZ, R. J. G. B**.; Curvas Elípticas aplicadas à Criptografia**. In: ERCEMAPI, 2011, Teresina. Encontro Regional Ceará Maranhão Piauí, 2011.

MCCUNE, Jonathan M.. et al. **Seeing-is-believing**: using camera phones for human-verifiable authentication. In: SECURITY AND PRIVACY, 2005, Oakland. Oakland: IEEE Computer Society, 2005. p.110-124.

NICOLAS, Stephane. **Robospice**. 2013. Disponível em: <https://github.com/octo-online/robospice>. Acesso em: 06 jan. 2014.

OREILLY, Eismann. **QR Codes Versus Data Matrix**. 2011. Disponível em: <http://qrworld.wordpress.com/2011/09/26/qr-codes-versus-data-matrix/>. Acesso em: 26 dez. 2013.

PARK, KW; LIM, S. e PARK, KH. **Computationally efficient PKI-based single sign-on protocol**, PKASSO for mobile devices. Computers, IEEE Transactions on, v. 57, n. 6, p. 821–834, 2008.

SQUARE INC. **Picasso**: A powerful image downloading and caching library for Android. 2013. Disponível em: <http://square.github.io/picasso/>. Acesso em: 06 jan. 2014.

SQUARE INC. **Retrofit**: A type-safe REST client for Android and Java. 2013. Disponível em: <http://square.github.io/retrofit/>. Acesso em: 06 jan. 2014.

STEVENSON, Alastair. **Android SecureRandom Bitcoin wallet vulnerability could be used to hack more than 300,000 apps**. 2013. Disponível em: <http://www.v3.co.uk/v3-uk/news/2288778/android-securerandom-bitcoin-wallet-vulnerability-could-be-used-to-hack-more-than-300-000-apps>. Acesso em: 26 dez. 2013.

TANVI, Parekh; SONAL, Gawshinde; KUMAR, Sharma Mayank. **Token based authentication using mobile phone**. In: COMMUNICATION SYSTEMS AND NETWORK TECHNOLOGIES, 2011, Katra, Jammu. Katra, Jammu: IEEE Computer Society, 2011. p.85-88.

THANH, Do Van. et al. **Strong authentication with mobile phone as security token**. In: MOBILE ADHOC AND SENSOR SYSTEMS, 6., 2009, Macau. Macau: IEEE Computer Society, 2009. p.777-782.

TYLEY, Roberto. **Spongy Castle**: repackage of Bouncy Castle for Android. 2013. Disponível em: <http://rtyley.github.io/spongycastle/>. Acesso em: 06 jan. 2014.

US DEPARMENT OF DEFENSE. **Globalplatform**: department of defense common access card case study. 2006. Disponível em: <http://www.globalplatform.org/uploads/DoD_Case_Study.pdf>. Acesso em: 02 jun. 2013.