UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
CENTRO MULTIDISCIPLINAR PAU DOS FERROS
DEPARTAMENTO DE ENGENHARIAS E TECNOLOGIA
CURSO DE GRADUAÇÃO EM TECNOLOGIA DA INFORMAÇÃO

ADBY DA SILVA SANTOS

APLICATIVO UNIVERSITÁRIO PARA ASSINATURAS DIGITAIS E AUTENTICAÇÃO

PAU DOS FERROS – RN
2020

ADBY DA SILVA SANTOS

APLICATIVO UNIVERSITÁRIO PARA ASSINATURAS DIGITAIS E AUTENTICAÇÃO

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Tecnologia da Informação do Departamento de Engenharias e Tecnologia da Universidade Federal Rural Do Semi-Árido, como requisito parcial à obtenção do grau de Bacharel em Tecnologia da Informação.

Orientador: Prof. Me. Marco Diego Aurélio Mesquita

Co-Orientador: Prof. Dr. Reudismam Rolim de Sousa

PAU DOS FERROS – RN

2020

S237a Santos, Adby da Silva.
Aplicativo Universitário para Assinaturas Digitais e Autenticação / Adby da Silva Santos. - 2020.
29 f. : il.

Orientador: Marco Diego Aurélio Mesquita.
Coorientador: Reudismam Rolim de Sousa.
Monografia (graduação) - Universidade Federal Rural do Semi-árido, Curso de Tecnologia da Informação, 2020.

1. Criptografia. 2. Assinatura Digital. 3. Android. 4. Aplicativo. I. Mesquita, Marco Diego Aurélio, orient. II. Sousa, Reudismam Rolim de, co-orient. III. Título.

ADBY DA SILVA SANTOS

APLICATIVO UNIVERSITÁRIO PARA ASSINATURA DIGITAL E AUTENTICAÇÃO

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Tecnologia da Informação do Departamento de Engenharias e Tecnologia da Universidade Federal Rural Do Semi-Árido, como requisito parcial à obtenção do grau de Bacharel em Tecnologia da Informação.

Aprovada em: 07 /02/2020

BANCA EXAMINADORA

Marco Diego Aurélio Mesquita

Prof. Me. Marco Diego Aurélio Mesquita (UFERSA)
Presidente

Prof. Dr. Claudio Andrés Callejas Olguín (UFERSA)
Membro Examinador

Pedro Thiago Valério de Souza

Prof. Me. Pedro Thiago Valério de Souza (UFERSA)
Membro Examinador

Dedico este trabalho a minha família, aos meus amigos e a Deus.

AGRADECIMENTOS

Agradeço primeiramente a Deus, Aquele que É, pela Graça e pela determinação concebida a mim. Louvado seja!

A minha família, em especial, meus pais, Lailton e Lenice, por todo apoio, amor, carinho e orações; são pais por excelência; meus primeiros professores; escolhidos por Deus. A casa familiar é um templo onde a paz divina reina e somos eternas crianças.

A minha amada Avó Francinete (*in memorian*) e ao meu querido Avô Moacir, pelos exemplos de fé, amor e dedicação à família.

Aos meus irmãos Ally "Lygon", sempre carregado no Dota, e Allyce; aos meus amigos de Jardim, da eterna 11-12; aos meus *confrades* do Fraternidade Libertária no Discord, em especial ao Vitor "*adamska"* Gomes, grande irmão na fé e no Dota; aos companheiros de batalha, em especial, Lucas "Lusca" Lima, Ernandes Maia, Igor "Raigor" Ramon, como também Jadson "Bel" Batista, David Dutra, Fernando Dutra e Jorge Edson. Aos companheiros Thiago Pereira e Gabriel Franco, Lords de AllProsia. Obrigado pela consideração e, acima de tudo, pelas risadas, pois o humor é definitivamente indispensável ao ser humano.

Ao meu orientador Marco e co-orientador Reudisman, pela paciência e contribuições fundamentais a este trabalho; aos demais professores que conheci na UFERSA; entre eles, destaco: Helder Oliveira, Claudio Callejas e Otávio Lavor.

Obrigado a todos que torceram direta e indiretamente. Que a Graça vos toque e vos consagre.

“Nada é mais trágico para um indivíduo que foi sábio outrora, do que perder sua memória e tradição.”

(Venerável Fulton Sheen)

RESUMO

Com a popularização dos sistemas digitais, principalmente a Internet, a comunicação pode ser ampliada a níveis globais. No entanto, um problema antigo persiste: a autenticidade das informações. Assim, técnicas de Criptografia, como a Assinatura Digital, foram desenvolvidas de forma a preservar a legitimidade das informações no contexto digital. Apesar de sua popularidade, em ambientes universitários cujos processos burocráticos e ineficazes perpetuam-se com a utilização de aparatos físicos, essas técnicas ainda não são amplamente utilizadas. Em 2019, constatou-se que cerca de 67% da população brasileira utilizava *smartphones*, ao passo que 38% usava *desktops* e *laptops*. Desses 67%, cerca de 85% utilizava o sistema operacional Android. Nesse sentido, este trabalho intentou projetar um sistema capaz de fornecer chaves, assinar arquivos e verificar arquivos rapidamente e, a partir dele, criar um aplicativo para Assinatura Digital disponível para dispositivos Android que possa ser utilizado cotidianamente. Ao término do trabalho, foi possível desenvolver um aplicativo capaz de gerar pares de chaves, assinar arquivos e verificar arquivos.

Palavras-chave: Criptografia, Assinatura Digital, Android, Aplicativo.

ABSTRACT

Because of the popularization of digital systems, mainly the Internet, communication can be expanded at global levels. However, an old problem persists: the authenticity of the information. Thus, Cryptography techniques, such as Digital Signature, were developed in order to preserve the legitimacy of information in the digital context. Despite their popularity, in university environments whose bureaucratic and ineffective processes are perpetuated with the use of physical devices, these techniques are not yet widely used. In 2019, it was found that about 67% of the Brazilian population used smartphones, while 38% used desktops and laptops. Among these 67%, about 85% used the Android operating system. Therefore, this work intends to design a system capable of providing keys, signing files and verifying files quickly and, from it, create an application for Digital Signature available for Android devices that can be used daily. Thereafter, it was possible to develop an application capable of generating key pairs, signing files and verifying files.

Keywords: Cryptography, Digital Signature, Android, App.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

Desde os tempos mais remotos, como no Egito Antigo, indivíduos intentam comunicar-se em segurança com outros, seja em um mesmo território, seja a quilômetros de distância (PAAR; PELZL, 2010). Considerando a distância e os problemas de autenticidade que tal comunicação poderia sofrer, foi necessário desenvolver técnicas engenhosas para garantir a titularidade correta não só na comunicação, mas também sobre riquezas e territórios, e é nesse contexto que surge a Criptografia.

Com o passar dos anos, inúmeras ideias surgiram. Dentre elas, a Assinatura Digital, idealizada na segunda metade do século XX, descreve-se como uma tecnologia responsável por manter a autenticidade de um determinado documento, garantindo que seu destinatário tenha plena noção da origem do documento. Essa tecnologia se popularizou de forma assustadora e, hoje, é largamente utilizada, desde em sites de *e-commerce* até aplicativos de bate-papo (WHATSAPPINC, 2016).

1.1 JUSTIFICATIVA

Não obstante, o setor público brasileiro em geral não utiliza de tecnologias nesse âmbito; não digitaliza seus processos. Boa parte dos processos da Universidade Federal Rural do Semi-Árido (UFERSA) se encaixa nesse contexto. As assinaturas de docentes, por exemplo, ainda hoje são feitas à mão.

O não uso de tecnologias como essas revela um atraso e uma consequente ineficiência nos processos de uma instituição. Não só isso, como também chances maiores de problemas como adulteração de documentos e repudiação acontecerem.

1.2 MOTIVAÇÃO

Cerca de 38% da população adulta brasileira utiliza computador (*desktop* ou *laptop*), ao passo que 67% utiliza *smartphone* (KEMP, 2019). Além disso, cerca de 85% destes utilizam sistema operacional (SO) Android (STATCOUNTER GLOBAL STATS, 2019). Considerando

isso e tendo em vista um maior alcance, o projeto visa desenvolver um protótipo na forma de um aplicativo para dispositivos móveis com SO Android.

Uma forma de amenizar tal problema é criar um sistema que garanta a privacidade e a autenticidade das informações. Dito isso, o presente trabalho terá, como fim, projetar uma aplicação responsável tanto por assinar digitalmente documentos de associados à UFERSA quanto por certificá-los.

Com a finalização do aplicativo, será possível aproveitar dos benefícios de Assinatura Digital, como a não repudiação e a não adulteração de documentos, bem como da praticidade e acessibilidade dos dispositivos móveis.

1.3 OBJETIVOS

1.3.1 Objetivo Geral

Desenvolver um aplicativo móvel para Android que garanta autenticidade aos objetos e agentes envolvidos em uma comunicação digital.

1.3.2 Objetivos Específicos

(a) Analisar tecnologias performáticas e confiáveis relacionadas à Assinatura Digital;

(b) Projetar um sistema que englobe Assinatura Digital disponível para dispositivos móveis Android.

(c) Desenvolver um protótipo de aplicativo que permita de forma simples assinar digitalmente um documento e verificar uma assinatura.

1.4 ESTRUTURA DO TRABALHO

Este trabalho se desenvolve com a seguinte organização: na seção **Fundamentação Teórica**, é feita uma revisão da literatura englobando os principais temas, conceitos e

ferramentas essenciais para a construção do trabalho; na seção **Desenvolvimento da Aplicação**, discute-se o desenvolvimento da aplicação proposta por este trabalho, descrevendo a metodologia, assim como as ferramentas e a abordagem adotada; na Seção **O Aplicativo Desenvolvido**, explana-se o produto de *software* descrito na Seção anterior; na Seção **Considerações Finais**, discute-se os resultados, considerações, limitações e trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Nesta seção serão discutidos conceitos fundamentais relacionados ao tema deste trabalho tais como Criptografia, Criptologia, Criptografia de Chave Pública, funções de *hash* e Assinatura Digital.

A partir desse ponto, é possível introduzir os conceitos de Criptografia Simétrica e Assimétrica e, assim, um exemplo passo a passo do algoritmo RSA (Rivest-Shamir-Adleman) é feito. Posteriormente, apresentam-se funções de *hash*, seus usos e características. O conhecimento até aqui relacionado nos permite então descrever o processo de assinatura digital. Por fim são citadas as ferramentas que foram usadas no desenvolvimento desse trabalho: O sistema operacional (SO) Android; O Ambiente de Desenvolvimento Integrado (em inglês, *Integrated Development Environment* – IDE) Visual Studio Code (VSC); a linguagem de programação Dart e o *framework* Flutter.

2.1 CRIPTOGRAFIA

Criptografia, de acordo com o Cambridge *Dictionary*, é a prática de criar e entender códigos que mantenham informação em segredo. Segundo Diffie e Hellman (1976), a criptografia busca resolver dois tipos de problemas de segurança: privacidade e autenticação. O primeiro está ligado à ideia de que uma mensagem, enviada por um canal de comunicação não seguro, pode somente ser lida pelo seu destinatário. Já autenticação subdivide-se em autenticação de mensagens, onde verifica-se se uma mensagem foi enviada por realmente quem disse ter enviado, e autenticação de usuário, a qual verifica se um determinado usuário é realmente quem diz ser. Nesse contexto, canal de comunicação não seguro não se refere a garantia de não-corrupção ou entrega da mensagem, entende-se que as informações transferidas através desse canal podem ser interceptadas por terceiros. No entanto, caso a aplicação de

criptografia seja bem-sucedida, essa interceptação não comprometerá o conteúdo da mensagem, isto é: embora um terceiro possa ter acesso à mensagem criptografada, não será possível compreendê-la.

Existem evidências históricas que em torno de 2000 a.C. os antigos Egípcios já utilizavam hieróglifos secretos. Técnicas de mensagem cifrada foram tecnologias presentes em milhares de culturas. Como exemplos históricos de criptografia, é possível citar a Scytale e a Cifra de César. Na Grécia, por volta de 250 a.C, a Scytale foi descrita por Plutarco. Em Roma, em 50 a.C, a Cifra de César foi empregada em atividades militares (PAAR; PELZL, 2010).

Vale notar a importância que a privacidade e o sigilo de informações têm, *e.g.,* liberdade de expressão, liberdade de associação, contratos. Em governos tirânicos, por exemplo, é comum que ocorram rastreios nas comunicações a fim de encontrar indivíduos ou grupos opositores para, assim, tomar medidas repressivas contra estes (FISHER, 2019). Dito isso, existem aplicações atuais que trabalham de forma a dificultar o rastreio de informações e de dados pessoais. O aplicativo mensageiro Telegram criptografa as mensagens de seus usuários de forma que somente estes tenham acesso direto às mensagens. Dessa forma, questões como sigilo da informação e privacidade dos usuários são frisadas (TELEGRAM, 2020).

2.2 CRIPTOLOGIA E CRIPTOANÁLISE

A criptologia pode ser definida como a ciência e, ainda, a arte de manipular sistemas de criptografia. A criptoanálise, que assim como a criptografia, é derivada da criptologia, estuda a quebra de códigos secretos (PAAR; PELZL, 2010). Dessa forma, esse tipo de estudo também pode ser utilizado para classificar os sistemas de criptografia com relação a seu potencial de fornecer segurança. A partir da criptoanálise, novos algoritmos de criptografia podem ser desenvolvidos e também desencorajar o uso de técnicas e algoritmos inseguros relacionados à criptografia, tal como o MD5, que se tornou inseguro com a evolução do poder computacional (MICROSOFT, 2013). Apesar de, considerando certos casos – como, por exemplo, o uso malicioso – aparentar ser uma atividade nociva, a criptoanálise é de suma importância (MCLURE; SCAMBRAY; KURTZ, 2012).

Como exemplo de uma aplicação histórica relevante de criptoanálise, pode-se citar os esforços de Turing e outros. Durante a Segunda Guerra Mundial, Turing, em Bletchley Park, foi um dos responsáveis por quebrar o código da Enigma, uma máquina criptográfica usada pelas forças armadas alemãs. Sua invenção, denominada Bombe, foi capaz de reduzir

significativamente o trabalho da equipe de quebradores de código e, com isso, prever ações advindas do inimigo mais facilmente (GANDY; YATES, 2001).

Após a Segunda Guerra Mundial, a ciência foi capaz de prosperar em várias partes do mundo e, assim, diferentes algoritmos de criptografia surgiram. Esses algoritmos podem ser classificados em duas categorias: a Criptografia Simétrica e a Criptografia Assimétrica (PAAR; PELZL, 2010).

2.3 CRIPTOGRAFIA SIMÉTRICA E ASSIMÉTRICA

Na Criptografia Simétrica uma mesma chave é usada tanto para criptografar quanto para descriptografar uma mensagem, ou seja, a forma de utilização da chave na Criptografia Simétrica é análoga ao uso da chave em uma porta, em que a mesma chave é usada para abrir ou fechar a porta. Já na Criptografia Assimétrica, para que dois interlocutores possam se comunicar bidirecionalmente de forma privada, é necessário criar um par de chaves para cada um dos usuários. Assim, ambos os processos de criptografia e descriptografia são, de uma forma geral, mais rápidos na Criptografia Simétrica do que na Criptografia Assimétrica. (PAAR; PELZL, 2010).

Na Criptografia Assimétrica, também denominada Criptografia de Chave Pública, há um par de chaves: uma chave pública k_{pub} e uma chave privada k_{pr}. Uma mensagem criptografada com a chave pública só pode ser descriptografada com a chave privada. A chave pública é conhecida abertamente e a chave privada é mantida em segredo. Pode-se fazer uma analogia da Criptografia Assimétrica com uma caixa de correio, onde, usualmente, qualquer pessoa pode inserir uma carta, mas somente o dono da caixa pode abri-la.

Há diferentes algoritmos de Criptografia Assimétrica, dentre eles pode-se citar o RSA, que será descrito na subseção seguinte, o Digital Signature Algorithm (DSA), criado pelo governo federal dos Estados Unidos da América, e *Elliptic Curves Digital Signature Algorithm* (ECDSA), que se baseia no problema do logaritmo discreto (PAAR; PELZL, 2010).

2.3.1 Algoritmo RSA

O algoritmo RSA foi desenvolvido por Rivest, Shamir e Adleman em 1978 com base em obras como o sistema de troca de chaves de Diffie e Hellman (*Diffie-Hellman key exchange* –

DHKE). A segurança desse algoritmo é baseada na dificuldade de se fatorar o produto resultante de dois números primos grandes (RIVEST, R. L; SHAMIR, A; ADLEMAN, L, 1978).

Em um possível processo de criptografia e descriptografia com RSA, a mensagem é criptografada com uma chave privada. Este processo gera uma mensagem cifrada. A mensagem cifrada é, então, descriptografada com a chave pública correspondente à chave privada. Vale lembrar que, assim como todo algoritmo de Criptografia Assimétrica, é necessário antes criar um par de chaves, i.e., chave pública e chave privada (STEYN, 2012).

O processo, então, pode ser descrito da seguinte forma:

1. São escolhidos dois números primos p e q.
2. Calcula-se o módulo $n = p * q$.
3. Calcula-se o totiente de n, denotado por $\phi(n)$, da seguinte forma: $(p-1)*(q-1)$.
4. Seleciona-se um número inteiro positivo aleatório e para assumir o papel de chave pública. Para isso, é necessário que e e $\phi(n)$ sejam primos entre si, ou seja, que tenham o número 1 como MDC. Ademais, e deve ser menor do que $\phi(n)$.
5. Determina-se d, que assumirá o papel de chave privada. Para isso, calcula-se o inverso multiplicativo de e módulo $\phi(n)$ através do Algoritmo de Euclides estendido.
6. Seleciona-se uma mensagem m.
7. Criptografa-se m a partir do seguinte cálculo: $m^e \bmod n$. Este processo gera a mensagem cifrada c.
8. Descriptografa-se c a partir do seguinte cálculo: $c^d \bmod n$.

Embora o exemplo chame d de chave privada e e de chave pública, para o passo de criptografia, usa-se e e n; enquanto para descriptografar, são usados d e n. Na prática, o par (e, n) é usado como chave pública e o par (d, n) é usado como chave privada. Com efeito, para números primos suficientemente grandes, obter d a partir de e e n é uma tarefa difícil. Ou seja: é relativamente fácil criptografar uma mensagem com a chave pública, mas muito difícil descriptografá-la sem a chave privada.

Exemplo:

1. Tem-se $p = 11$ e $q = 13$;
2. Calcula-se o módulo $n = 11*13 = 143$;
3. Calcula-se o totiente de n: $\phi(n) = (11-1)*(13-1) = 120$;
4. Seleciona-se a chave pública $e = 7$;
5. Determina-se a chave privada $d = 103$, que satisfaz a condição de $[(7*103) \bmod 120] = 1$;

6. Tem-se a mensagem $m = 9$;
7. Criptografa-se m: $9^7 \bmod 143 = 48 = c$.
8. Descriptografa-se c: $48^{103} \bmod 143 = 9$.

Vale notar que caso a mensagem do remetente fosse criptografada com a chave pública do destinatário, a chave utilizada para descriptografar a mensagem cifrada seria a chave privada do destinatário.

O algoritmo RSA e os valores de exemplo foram escolhidos para ilustrar Criptografia Assimétrica apenas por razões didáticas. Na prática, números primos muito maiores devem ser utilizados.

Poder-se-ia atentar para o fato de que mensagens nem sempre são números como no exemplo acima. Todavia, qualquer informação discreta pode ser representada por um número ou por uma sequência de números. Vale também notar que potências de números grandes podem ser demasiadamente complicadas de se calcular; um uso prático de RSA pode, por exemplo, usar Criptografia Assimétrica apenas para negociar de forma segura uma chave para uso posterior de Criptografia Simétrica.

2.3.2 Funções *Hash*

Funções *hash* são algoritmos criptográficos que, a partir de uma mensagem de entrada, geram uma saída resumida com número fixo de *bits*. Tal processo é bastante conveniente para o uso de criptografia, uma vez que garante uma entrada generalizada, de tamanho fixo (geralmente curta) e de difícil compreensão por terceiros. Funções *hash* são utilizadas no processo de assinatura digital, em conjunto com algoritmos de Criptografia Assimétrica, tal como o RSA (PAAR; PELZL, 2010).

Devido a suas características, as funções *hash* são bastante usadas para promover a integridade de dados. Ao baixar um arquivo da Internet, tal como um sistema operacional, disponibiliza-se um código *hash* que o usuário pode usar para verificar se os dados dos arquivos não foram corrompidos durante a transmissão.

Colisão, no contexto de funções *hash*, é uma situação onde duas entradas diferentes x e y possuem o mesmo código *hash*, i.e., $H(x) = H(y)$. Uma das características de uma função *hash*, é que ela deve ser resistente a colisão (PAAR; PELZL, 2010). Diz-se que uma função *hash* é resistente à colisão quando é computacionalmente inviável encontrar outro dado que produza como saída o mesmo código *hash*. No entanto, considerando todas as possíveis entradas de

todos os possíveis tamanhos, evitar uma colisão é impossível, o que faz com o que os melhores algoritmos sejam aqueles capazes de tornar essa tarefa árdua o suficiente para ser considerada computacionalmente inviável (AJTAI, 1992).

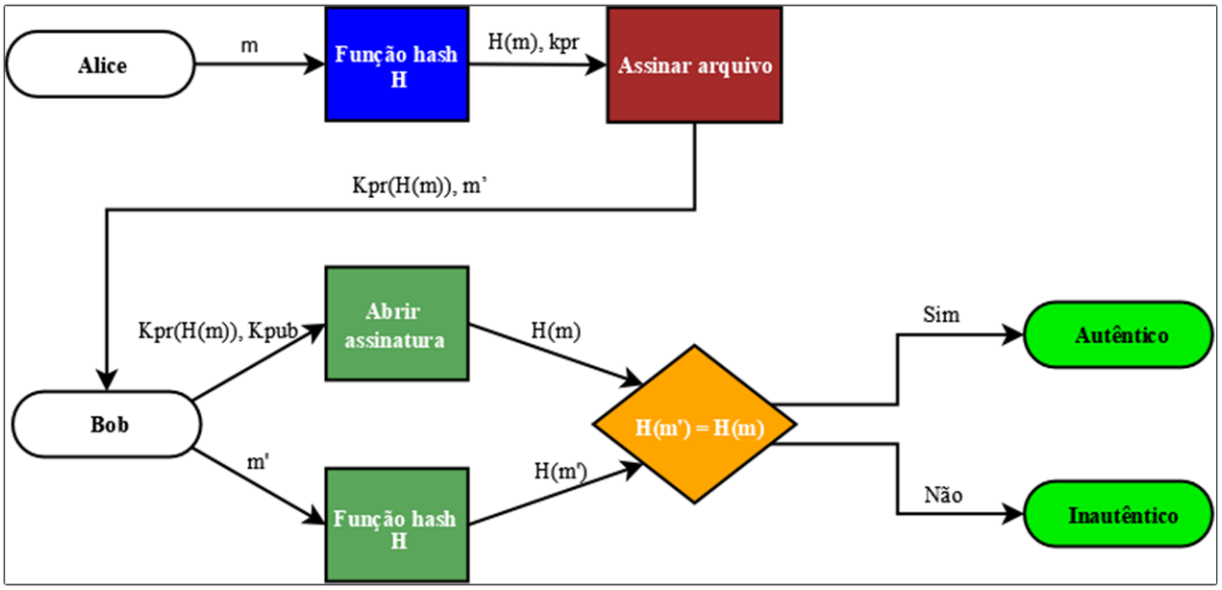
Assim como os algoritmos de Criptografia Assimétrica, existem diferentes funções *hash*, dentre elas o algoritmo SHA-1, desenvolvido pelo NIST (NIST, 2015), que produz uma saída de 160 *bits* a partir de uma mensagem de até 2^{64} *bits* de largura máxima (quantidade de *bits* da mensagem) (PAAR; PELZL, 2010). A mensagem é transformada em pedaços de 512 *bits* que segue um processo de 80 fases, o qual é dividido em 4 estágios de 20 fases cada (PAAR; PELZL, 2010). No entanto, o algoritmo SHA-1, com a evolução do poder de computação já é considerado vulnerável (WANG, X; YIN, Y. L; YU, H, 2005; MICROSOFT, 2015). Outro algoritmo de *hash* é o MD5, que produz uma saída de 128 *bits* e foi largamente utilizado no passado e que atualmente é considerado inseguro (MICROSOFT, 2012).

Entre os algoritmos de *hash* resistentes aos testes de colisão está o SHA-256, que faz parte do conjunto de algoritmos da família SHA-2 (NIST, 2015). O SHA-256 é capaz de produzir uma saída de 256 *bits*, diferente do SHA-1, cuja saída é de 160 *bits*. Apresentado pela primeira vez há quase 20 anos, o SHA-256 ainda é largamente usado e, apesar da crescente computação quântica, ainda é considerado seguro (MAVROEIDIS et al. 2018).

2.3.3 Assinatura e Autenticação

Em síntese, um possível processo de assinatura digital ocorre da seguinte forma: Alice quer enviar uma mensagem *m* para Bob. Para isso, Alice transforma *m* em código *hash* a partir de uma função *hash* H, logo, H(*m*). Em seguida, Alice criptografa H(*m*) com sua chave privada k_{pr}, logo, k_{pr}(H(*m*)). Por fim, Alice envia dois documentos: Uma cópia da mensagem original *m*' e a mensagem criptografada k_{pr}(H(*m*)). Bob recebe, então, dois arquivos. De posse da chave pública k_{pub} de Alice, Bob descriptografa k_{pr}(H(*m*)) e, assim, tem em mãos H(*m*). Em seguida, Bob aplica a função H na cópia da mensagem original *m*' (logo, H(*m*')) e compara ambos H(*m*) e H(*m*'). Caso o resultado seja o mesmo, a assinatura é autêntica; caso contrário, inautêntica (PAAR; PELZL, 2010). A Figura 1 ilustra esse exemplo.

Figura 1 - Da assinatura à autenticação



Fonte: Autoria própria, 2020

É importante esclarecer que a função de *hash* escolhida tem um impacto significativo sobre a segurança da assinatura. Caso seja fácil conseguir colisões de *hash*, o algoritmo de *hash* em questão não é adequado uma vez que tornaria viável a falsificação de assinaturas. O SHA-256 é um exemplo algoritmo de *hash* considerado seguro hoje (MAVROEIDIS et. al, 2018).

2.4 ANDROID

Android é um sistema operacional da Google para dispositivos móveis, como *smartphones* e *tablets*. No ano de 2019, constatou-se que Android é o sistema operacional móvel mais utilizado do mundo (OVIDE, 2019).

A Google disponibiliza o aplicativo Play Store, o qual é uma espécie de *marketplace* sob controle dela, onde aplicativos dos mais diversos tipos podem ser publicados por usuários, *e.g.*, editores de texto, jogos, *e-mail* eletrônico. Devido sua visibilidade, esse espaço é usado por desde empresas privadas, como Microsoft e Facebook, até instituições governamentais.

Assim, decidiu-se utilizar Android por ele ser amplamente utilizado e por apresentar um ambiente de produção e desenvolvimento de aplicativos amigável e acessível.

2.5 DART E FLUTTER

Dart é uma linguagem de programação multiplataforma e de código aberto mantida pela Google lançada oficialmente em 2011. Comparada à linguagem de programação JavaScript, Dart apresenta melhor desempenho (CODEMAGIC, 2019).

Com o anúncio do *framewok* Flutter para desenvolvimento de aplicativos móveis (*e.g.*, Android, iOS) em 2017, a popularidade dessa linguagem cresceu consideravelmente (CODEMAGIC, 2019). Mesmo sendo uma tecnologia relativamente nova, inúmeras empresas já adotam Flutter no desenvolvimento de suas aplicações; entre elas: Nubank, BMW, eBay, Tencent e Square. (GOOGLE, 2020).

Assim, decidiu-se utilizar Dart por ser uma linguagem em ascensão, pelo *framework* Flutter e pela organização que os mantém, no caso, a Google. Ambos Dart e Flutter estão disponíveis sob a licença BSD-*3-Clause*[1][2].

2.5.1 Pointy Castle

Pointy Castle é uma biblioteca Dart mantida pelo desenvolvedor Steven Roose para criptografia e descriptografia. Baseada no projeto Bouncy Castle Java e disponível sob a licença MIT[3], essa biblioteca fornece as ferramentas necessárias para gerar pares de chaves (*e.g.* ECDSA, RSA), gerar assinaturas e verificar arquivos.

Dado o exposto, decidiu-se usar tal biblioteca após a constatação que esta corrobora ativamente com o trabalho aqui desenvolvido.

3 DESENVOLVIMENTO DA APLICAÇÃO

Na Subseção **Metodologia**, apresentam-se os algoritmos selecionados; na Subseção **Ferramentas Utilizadas**, comentam-se as ferramentas utilizadas que dão suporte aos algoritmos supracitados; na Subseção **Descrição do Projeto e Abordagem**, discorre-se em detalhes sobre o projeto e sobre a abordagem utilizada para construção dele; na Subseção **O**

[1] https://github.com/dart-lang/sdk/
[2] https://github.com/flutter/flutter
[3] https://github.com/bcgit/pc-dart

Aplicativo Desenvolvido, é apresentado o aplicativo desenvolvido, explicando sua interface e suas funcionalidades.

3.1 METODOLOGIA

Para atingir os objetivos deste trabalho, foi feita, inicialmente, uma revisão bibliográfica para definir os conceitos necessários ao entendimento da aplicação bem como seu escopo.

Assim, foi feita uma pesquisa a partir da fundamentação teórica, onde as ferramentas a serem utilizadas foram escolhidas e seguiu-se com uma implementação. O processo de pesquisa destacou termos como: *Cryptography*; *Asymmetric Cryptography*; *Public Key Cryptography*; *Digital Signature*; *Hash Functions*; *Android*; e termos correlatos. Com isso, livros, artigos e websites foram selecionados. Dentre eles, importante destacar o artigo *New Directions in Cryptography*, desenvolvido por W. Diffie e M. E. Hellman em 1976; o artigo *A Method for Obtaining Digital Signature and Public-Key Cryptosystems*, de R. L. Rivest, A. Shamir e L. A. Adleman, lançado em 1978, e o livro *Understanding Cryptography*, de C. Paar e J. Pelzl, publicado em 2010.

Com o escopo definido, foi possível escolher quais algoritmos seriam utilizados pela aplicação, especificamente, a função *hash* e o algoritmo de assinatura digital. Para este trabalho, foi utilizada uma função *hash* SHA-256 e um algoritmo RSA para assinatura digital.

Vale notar que a seleção de uma função *hash* ou algoritmo de assinatura diferente não causaria impacto significativo no código da aplicação, uma vez que a aplicação foi desenvolvida de maneira tal que este tipo de alteração pode ser facilmente implementado.

3.2 FERRAMENTAS UTILIZADAS

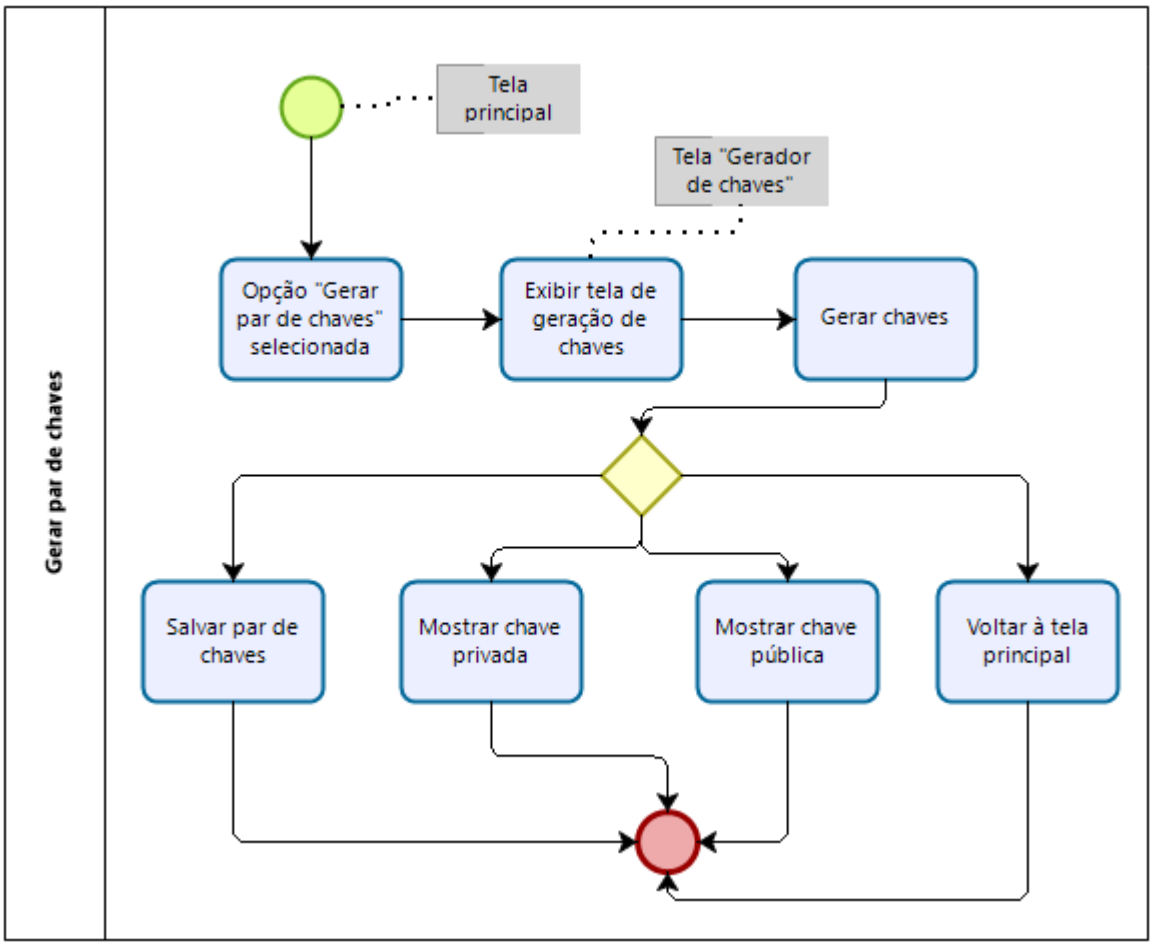
Foram utilizadas três bibliotecas auxiliares: *asn1lib*, para salvar as chaves geradas em formato padrão *Public-Key Cryptography Standards* (PKCS) (PALMA, 2019); *file_picker*, que viabilizou a seleção de arquivos; e a *path_provider*, que possibilitou a criação de diretórios e arquivos.

3.3 DESCRIÇÃO DO PROJETO E ABORDAGEM

Para este trabalho, foram definidas três funcionalidades principais para a aplicação: Gerar par de chaves; assinar arquivo; verificar arquivo.

A funcionalidade de gerar par de chaves funciona conforme o diagrama na Figura 2. A partir dele, entende-se que o usuário intenta criar uma par de chaves; com isso, o usuário seleciona a opção “Gerar par de chaves” na tela principal do aplicativo; assim, o aplicativo redireciona para uma nova tela denominada “Gerador de chaves”; nessa tela, o usuário seleciona a opção “Gerar par de chaves”; o aplicativo, então, inicia o processo de geração de um par de chaves, uma privada e uma pública (para assinar e para verificar arquivos, respectivamente); ao término desse processo, o usuário encontrará três novas opções: Mostrar chave privada; mostrar chave pública; e salvar par de chaves. De qualquer forma, o processo para gerar um par de chaves é concluído.

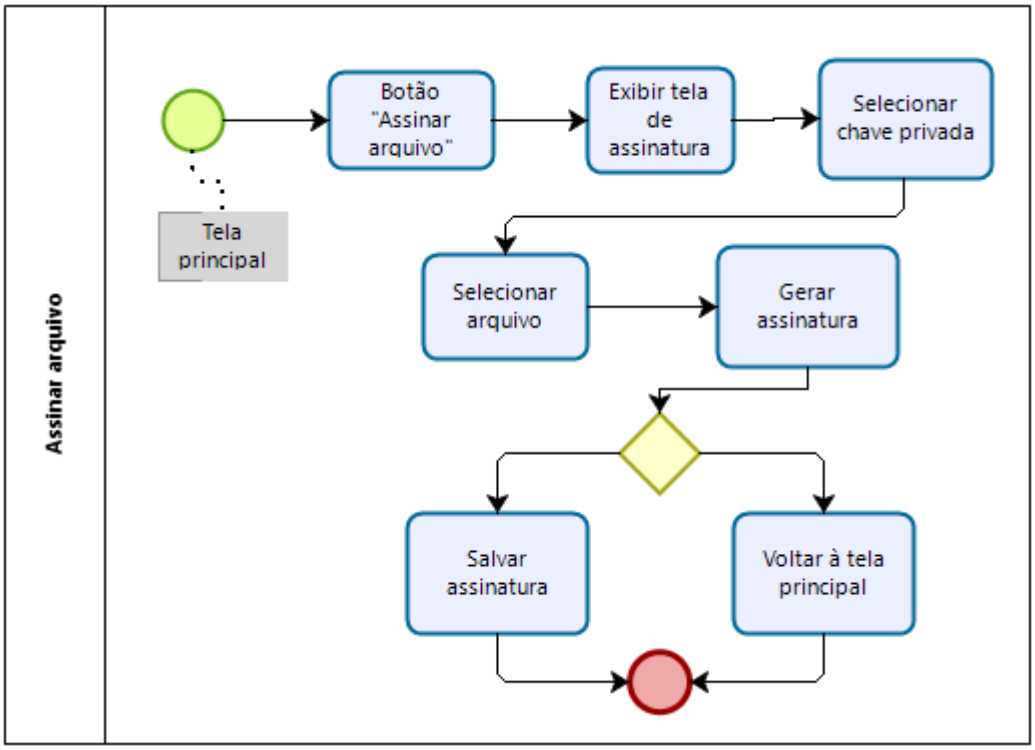
Figura 2 – Fluxo para geração de par de chaves



Fonte: Autoria própria, 2020

A funcionalidade de assinar arquivo é descrita pela Figura 3. Conforme descrito, o usuário, na tela principal do aplicativo, seleciona a opção “Selecionar chave privada”; assim, o aplicativo redireciona para uma tela específica para essa funcionalidade; em seguida, o usuário seleciona a opção “Selecionar chave privada”; após a seleção da chave privada, a opção “Selecionar arquivo” é selecionada; após a seleção do arquivo, o usuário seleciona a opção “Gerar assinatura”; assim, o processo de geração de assinatura digital do arquivo é iniciado; ao término desse processo, a opção “Salvar assinatura” é habilitada; de qualquer forma, o processo de assinatura de arquivo é concluído.

Figura 3 – Fluxo para assinar arquivo

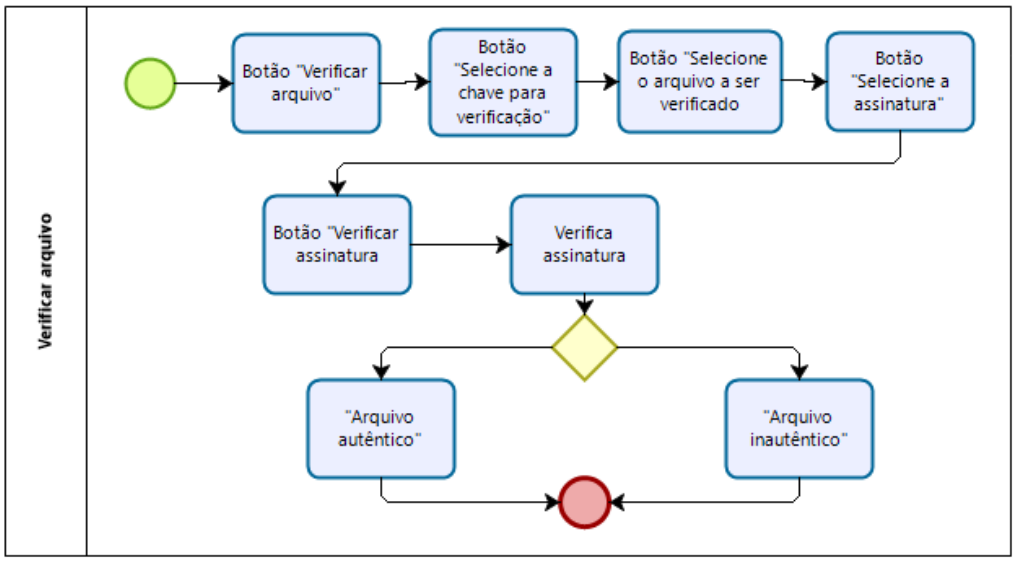


Fonte: Autoria própria, 2020

Já a funcionalidade de verificar arquivo funciona como ilustra a Figura 4. Assim, o fluxo inicia na tela principal do aplicativo; o usuário seleciona a opção “Verificar arquivo”; então, o aplicativo redireciona para uma tela específica de verificação; com isso, o usuário seleciona a opção “Selecione a chave para verificação” para escolher uma chave pública; concluída a seleção, o usuário deve selecionar um arquivo para ser verificado; em seguida, o usuário deve selecionar uma assinatura digital; assim, o usuário seleciona a opção “Verificar assinatura”; o

aplicativo, então, inicia o processo de verificação do arquivo fornecido; ao término da verificação, o resultado é mostrado; caso a assinatura coincida com o arquivo fornecido, o resultado apresentado é “Arquivo autêntico”; caso contrário, o resultado é “Arquivo inautêntico”.

Figura 4 – Fluxo para verificar arquivo



Fonte: Autoria própria, 2020

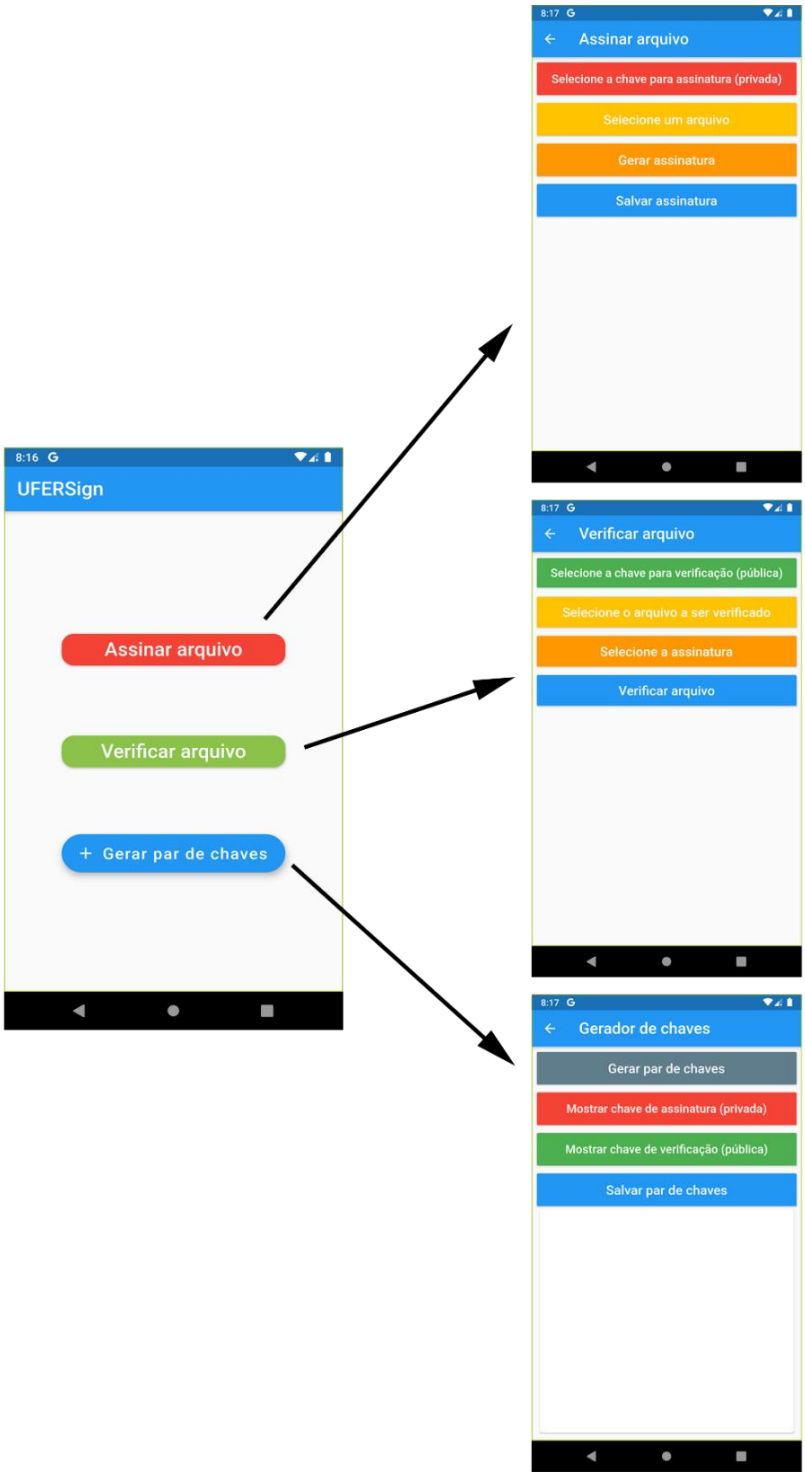
3.4 O APLICATIVO DESENVOLVIDO

Seguindo a descrição da subseção anterior, foi desenvolvida uma aplicação composta por quatro telas: Tela principal; tela de assinar arquivo; tela de verificar arquivo; e tela de geração de chaves. Vale frisar que todos os dados persistidos são salvos na memória do dispositivo em diretórios distintos para cada tipo de arquivo, de modo que, quando solicitado que o usuário selecione um arquivo, assinatura ou chave, o mesmo possa acessá-los através de uma interface de seleção do próprio SO Android. Além disso, a chave pública e a chave privada adotaram, respectivamente, os títulos de “chave de verificação” e “chave de assinatura” visando uma descrição expressiva e clara da função de cada chave. Dado o exposto, seguiu-se para a descrição das telas desenvolvidas.

A tela principal (Figura 5) é responsável por apresentar as funcionalidades ao usuário, de forma que, ao selecionar uma das três opções, o usuário seja redirecionado para a tela desejada.

A tela de assinar arquivo (Figura 5) é responsável por abarcar os seguintes comportamentos: Selecionar chave privada; selecionar arquivo; gerar assinatura; e salvar assinatura.

Figura 5 – Interfaces da aplicação

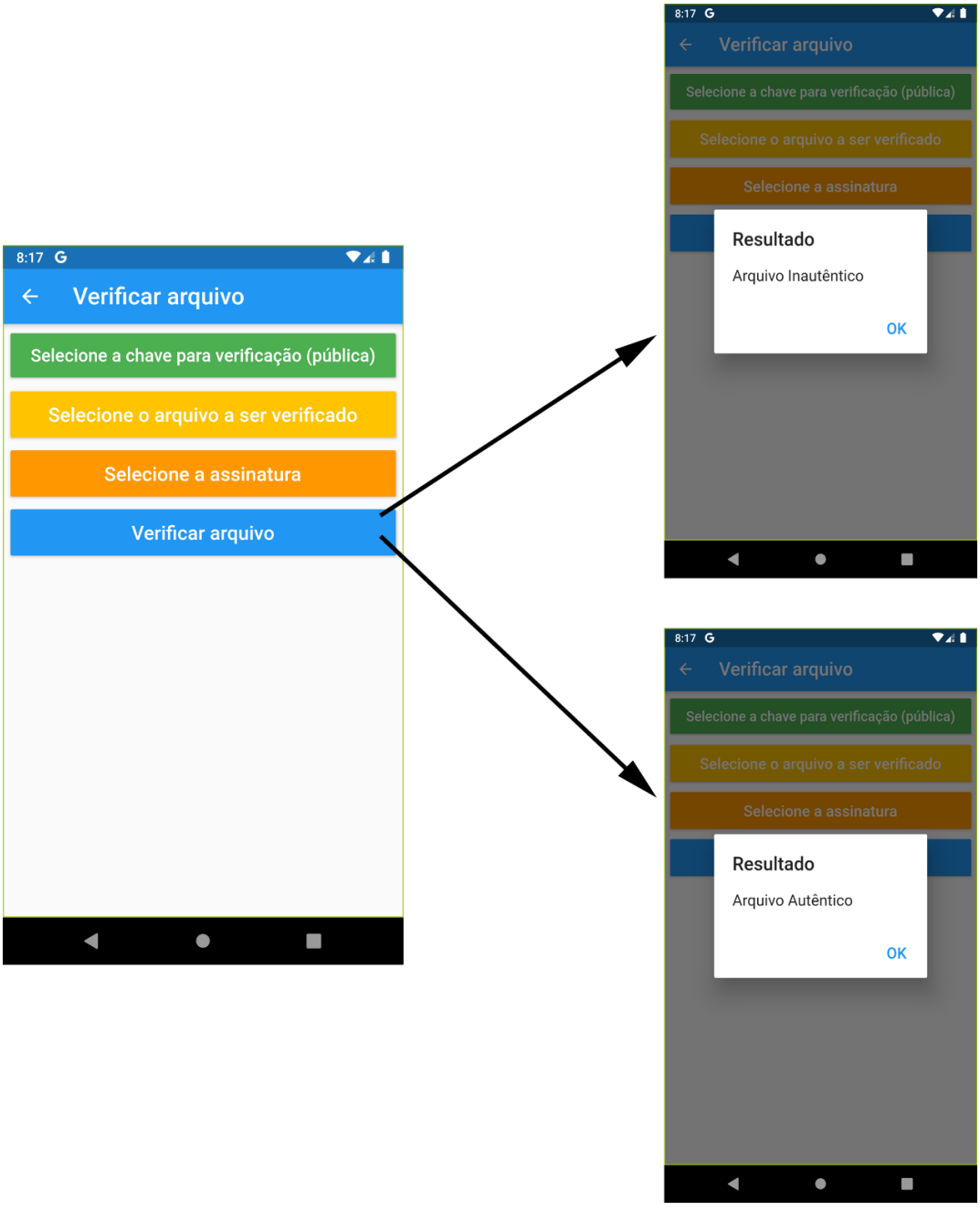


Fonte: Autoria própria, 2020

A tela de verificar arquivo (Figura 5), por sua vez, é responsável por conter as seguintes instruções: Selecionar chave pública; selecionar arquivo a ser verificado; selecionar assinatura; e verificar arquivo.

Na Figura 6, são apresentadas as possíveis saídas do processo de verificação de arquivo na aplicação.

Figura 6 – Saídas do processo de verificação de arquivo



Fonte: Autoria própria, 2020

A tela de geração de chaves (Figura 5) é encarregada de abarcar as seguintes instruções: Gerar par de chaves; apresentar chave pública; apresentar chave privada; e salvar o par de chaves. Uma mensagem de sucesso é apresentada após o sistema salvar o par de chaves. Vale notar que as opções “Mostrar chave de assinatura (privada)”, “Mostrar chave de verificação (pública)” e “Salvar par de chaves” somente estarão disponíveis caso o usuário selecione a opção “Gerar par de chaves” e esse processo suceda corretamente.

4 CONSIDERAÇÕES FINAIS

Hoje, diversas aplicações e sites utilizam de Assinatura Digital intentando garantir a autenticidade de seus usuários. Nesse sentido, buscou-se implementar uma aplicação que criasse pares de chaves e possibilitasse a assinatura e verificação de arquivos, de forma a auxiliar atividades cotidianas. Assim, foi feita uma pesquisa bibliográfica de forma a estabelecer um entendimento básico sobre o contexto de uma aplicação dessa natureza. Feita tal pesquisa, ferramentas e tecnologias foram selecionadas de forma a facilitar o desenvolvimento como também atender a qualidade da aplicação desejada.

Inicialmente, este trabalho contaria com a linguagem de programação Kotlin e eventuais ferramentas compatíveis com esse contexto. No entanto, algumas questões motivaram a mudança de ferramentas. Dentre elas, destacam-se: Curva de aprendizado menor; Flutter fornece muito material de *design* pronto; Maior transparência da biblioteca Pointy Castle em relação àquela fornecida pela Google.

No entanto, foram encontradas algumas limitações na aplicação. Entre elas, carência de interface intuitiva para seleção de chaves, assinaturas e arquivos; de opções de compartilhamento de arquivos, chaves e assinaturas; de entidade certificadora, a qual valide os usuários e suas respectivas chaves. Ademais, não foram feitos testes de validação com possíveis usuários.

4.1 TRABALHOS FUTUROS

Para trabalhos futuros, é sugerida a implementação de verificação dos arquivos, assinaturas e chaves selecionadas, uma vez que a implementação atual não verifica a procedência destes passos.

Outra sugestão é integrar o ambiente de certificação digital ao aplicativo, de forma que organizações privadas e/ou públicas verifiquem e aprovem as chaves utilizadas pelos usuários do aplicativo.

Com o objetivo de melhorar a usabilidade, pode-se implementar funcionalidades de criptografia, descriptografia e troca de chaves e dados criptografados ou assinados pelos usuários de forma mais intuitiva. Uma possibilidade pode ser, por exemplo, o uso de *bluetooth*, SMS ou, ainda, uma rede sem fio (Wi-Fi) como meio para troca.

Visando aumentar a diversidade de opções de algoritmos, poder-se-ia implementar algoritmos mais avançados, como o HMAC, utilizado pelo Whatsapp. (WHATSAPPINC, 2016).

REFERÊNCIAS

AJTAI, M. The complexity of the Pigeonhole Principle. **Combinatorica**, [*S. l.*], v. 14, n. 4, p. 417-433, 22 fev. 1992. DOI https://doi.org/10.1007/BF01302964. Disponível em: <https://www.ukessays.com/essays/mathematics/applications-of-the-pigeonhole-principle-mathematics-essay.php?vref=1>. Acesso em: 4 dez. 2019.

CODEMAGIC. **Dart vs JavaScript**. [S. l.], 5 mar. 2019. Disponível em: <https://blog.codemagic.io/dart-vs-javascript/>. Acesso em: 18 jan. 2020.

CRYPTOGRAPHY | meaning in the Cambridge English Dictionary. **Cambridge Dictionary | English Dictionary, Translations & Thesaurus**. Disponível em: <https://dictionary.cambridge.org/dictionary/english/cryptography>. Acesso em: 08 out. 2019.

DIFFIE, W; HELLMAN, M. New directions in cryptography. **IEEE transactions on Information Theory**, v. 22, n. 6, p. 644-654, 1976. Disponível em: <https://caislab.kaist.ac.kr/lecture/2010/spring/cs548/basic/B08.pdf>. Acesso em: 10 out. 2019.

FISHER, C. **FISA court: FBI use of NSA‘s electronic surveillance data was illegal**: The surveillance court found tens of thousands of improper database searches in 2017 and 2018. [S. l.], 8 out. 2019. Disponível em: <https://www.engadget.com/2019/10/08/fbi-surveillance-violated-constitutional-rights/>. Acesso em: 11 jan. 2020.

GANDY, R. O.; YATES, C. E. M. **Collected Works of A.M.Turing: Mathematical Logic**. 1. ed. [*S. l.*]: North Holland, 2001. 306 p. v. 4.

GOOGLE. **Showcase - Flutter**. [S. l.], 2020. Disponível em: <https://flutter.dev/showcase>. Acesso em: 19 jan. 2020.

KEMP, S. **Digital 2019: Brazil**. 31 jan. 2019. Disponível em: <https://datareportal.com/reports/digital-2019-brazil>. Acesso em: 3 de out. 2019.

MAVROEIDIS, V et al. The Impact of Quantum Computing on Present Cryptography. **International Journal of Advanced Computer Science and Applications**, [*S. l.*], ano 3, v. 9, p. 405-414, 31 mar. 2018. Disponível em: <https://thesai.org/Downloads/Volume9No3/Paper_54-The_Impact_of_Quantum_Computing.pdf>. Acesso em: 12 dez. 2019.

MCLURE, S.; SCAMBRAY, J.; KURTZ, G. **Hacking Exposed 7**: Network Security Secrets and Solutions. 7. ed. [*S. l.*]: McGraw-Hill Education, 2012.

MICROSOFT. **The Microsoft Extensible Authentication Protocol-Message Digest 5 (EAP-MD5) implementation is being deprecated from versions of Windows.** [*S. l.*], 16

mar. 2012. Disponível em: <https://support.microsoft.com/en-us/help/922574/the-microsoft-extensible-authentication-protocol-message-digest-5-eap>. Acesso em: 4 dez. 2019.

_____. **Microsoft Security Advisory: Update for deprecation of MD5 hashing algorithm for Microsoft root certificate program: August 13, 2013.** [*S. l.*], 13 ago. 2013. Disponível em: <https://docs.microsoft.com/en-us/security-updates/SecurityAdvisories/2014/2862973>. Acesso em: 4 dez. 2019.

_____. **Windows Enforcement of SHA1 Certificates.** [*S. l.*], 24 set. 2015. Disponível em: <https://social.technet.microsoft.com/wiki/contents/articles/32288.windows-enforcement-of-sha1-certificates.aspx>. Acesso em: 4 dez. 2019.

_____. **Visual Studio Code**. [S. l.], 2020. Disponível em: <https://code.visualstudio.com/>. Acesso em: 18 jan. 2020.

NIST. Information Technology Laboratory. FIPS PUB 180-4, de Agosto de 2015. **Federal Information Processing Standards Publication 180-4**, Gaithersburg, MD, 2015. Disponível em: <https://dx.doi.org/10.6028/NIST.FIPS.180-4>. Acesso em: 4 dez. 2019.

OVIDE, S. The Smartphone Revolution Was the Android Revolution. **Bloomberg Businessweek**, 6 ago. 2019. Disponível em: <https://www.bloomberg.com/graphics/2019-android-global-smartphone-growth/>. Acesso em: 11 de out. 2019.

PAAR, C; PELZL, J. **Understanding Cryptography: A Textbook for Students and Practitioners**. 1. ed. [*S. l.*]: Springer-Verlag Berlin Heidelberg, 2010. 372 p. Disponível em: <https://b-ok.cc/dl/611947/172797>. Acesso em: 10 out. 2019.

PALMA, G. **Asymmetric Key Generation in Flutter**. [*S. l.*], 10 mar. 2019. Disponível em: <https://medium.com/flutter-community/asymmetric-key-generation-in-flutter-ad2b912f3309>. Acesso em: 19 jan. 2020.

RIVEST, R. L; SHAMIR, A; ADLEMAN, L. A Method for Obtaining Digital Signature and Public-Key Cryptosystems, **Communications of the ACM**, vol. 21, No. 2, Fev. 1978. Disponível em: <https://people.csail.mit.edu/rivest/Rsapaper.pdf>. Acesso em: 10 de out. 2019.

STATCOUNTER GLOBAL STATS. **Mobile Operating System Market Share Brazil**. 2019. Disponível em: <https://gs.statcounter.com/os-market-share/mobile/brazil/2019>. Acesso em: 3 de out. 2019.

STEYN, B. **How RSA Works With Examples**. [*S. l.*], 26 maio 2012. Disponível em: <http://doctrina.org/How-RSA-Works-With-Examples.html>. Acesso em: 8 dez. 2019.

TELEGRAM.ORG. **Telegram F.A.Q**. [S. l.], 2020. Disponível em: <https://telegram.org/faq#q-how-secure-is-telegram>. Acesso em: 11 jan. 2020.

WANG, X.; YIN, Y. L.; YU, H. Finding Collisions in the Full SHA-1. **Advances in Criptology – CRYPTO 2005**, v. 3621, ed. 1, p. 17-36, 2005.

WANG, X; YU, H. How to Break MD5 and Other Hash Functions. **Advances in Criptology - EUROCRYPT 2005**, v. 3494, ed. 1, p. 19-35, 2005.

WHATSAPPINC. **WhatsApp Encryption Overview**: Technical white paper. [S. l.], 5 abr. 2016. Disponível em: <https://scontent.whatsapp.net/v/t61/68135620_760356657751682_6212997528851833559_n.pdf/WhatsApp-Security-White-paper.pdf?_nc_ohc=CHOBWPVCIIYAX-UjBbe&_nc_ht=scontent.whatsapp.net&oh=37c3f365f636ef84df192ff42edbd824&oe=5E373C3B>. Acesso em: 31 jan. 2020.