Instituto Politécnico de Leiria
Escola Superior de Tecnologia e Gestão
Departamento de Engenharia Informática
Mestrado em Cibersegurança e Informática Forense

PATTERNLOCKSCRAMBLER: IMPEDIR ATAQUES OVER-THE-SHOULDER AO BLOQUEIO POR PADRÃO

RICARDO PERPÉTUO CAÇÃO

Leiria, Setembro de 2023

Instituto Politécnico de Leiria
Escola Superior de Tecnologia e Gestão
Departamento de Engenharia Informática
Mestrado em Cibersegurança e Informática Forense

PATTERNLOCKSCRAMBLER: IMPEDIR ATAQUES OVER-THE-SHOULDER AO BLOQUEIO POR PADRÃO

RICARDO PERPÉTUO CAÇÃO
Número: 2213267

Projeto realizado sob orientação do Professor Ricardo Jorge Pereira Gomes (ricardo.p.gomes@ipleiria.pt) e Professora Doutora Sónia Maria Almeida da Luz (sonia.luz@ipleiria.pt).

Leiria, Setembro de 2023

AGRADECIMENTOS

Em primeiro lugar agradeço a todos os amigos, colegas e familiares que me acompanharam durante o meu percurso académico e contribuíram, direta ou indiretamente, para a conclusão de mais uma etapa da minha formação. Destaco os colegas Gil Aguilar e Ricardo Pêgo que sempre me acompanharam nesta jornada. Quero também agradecer à Luana Andrade por todo o apoio e motivação dado nesta caminhada.

RESUMO

A autenticação nunca foi tão importante no mundo como é agora. Não é só a autenticação em programas e *websites* que importa, no entanto, *Smartphones*, *tablets* e outros dispositivos móveis passaram de ser apenas assistentes pessoais, tornando-se predominantes com funções muito mais poderosas com acesso a inúmeros serviços previamente só disponíveis em computadores fixos. Com o acesso a todos esses serviços e funcionalidades, os *smartphones* são agora cofres onde se guardam todas as informações que mais se precisam ter à mão, sejam estas informações pessoais e sensíveis ou não. No entanto, esta comodidade acarreta riscos consideráveis em termos de segurança e privacidade. Os *smartphones* contêm pormenores íntimos das nossas vidas: com quem falamos e com quem passamos tempo, aonde vamos e, cada vez mais, como gastamos o nosso dinheiro. A autenticação dos smartphones então, é essencial devido a exatamente esse mesmo aumento do uso de dispositivos pessoais que certamente armazenam informação pessoal e/ou sensível.

Cada maneira de proteger a sua informação tem pontos fortes e fracos, quer seja na usabilidade como na própria segurança do sistema protetor e é por isso que a procura de métodos mais avançados de autenticação ainda não acabou. Neste projeto realizou-se uma investigação abrangente sobre uma variedade de métodos de autenticação, considerando-se tanto aqueles de aplicação geral quanto aqueles específicos para *smartphones*. Foi analisado em específico o método de autenticação por desenho de padrão, identificada uma falha inerente a esse sistema, vulnerabilidade a ataques *over-the-shoulder*/*shoulder-surfing*.

Por fim foi construído um protótipo, denominado PatternLock Scrambler, que visa minimizar a eficácia de ataques *over-the-shoulder*. O protótipo utiliza mecanismos de transformação para receber um desenho de padrão vindo do utilizador e aplicar operações transformativas com objetivo de que um atacante que esteja a espiar a vítima não se aperceba que o padrão inserido não é o padrão autenticador para todos os casos.

Para validar a solução proposta foi realizado um questionário a utilizadores que permitiu concluir que o PatternLock Scrambler funciona quando empregue em aplicações com elevado grau de segurança e utilização menos regular.

ABSTRACT

Authentication has never been as important in the world as it is now. But it's not just authentication on computer programs and websites that matters, though. Smartphones, tablets, and other mobile devices are no longer personal assistants, but have become prevalent with much powerful functions and access to countless services previously only available on fixed computers.

With access to all these services and functionalities, smartphones are now vaults where you keep all the information you most need to have to hand, whether it's personal and sensitive information or not. However, this convenience comes with considerable security and privacy risks. Smartphones contain intimate details of our lives: who we talk to and spend time with, where we go and, increasingly, how we spend our money. Authentication of smartphones is therefore essential due to the very same increase in the use of personal devices that certainly store personal and/or sensitive information.

Each way of protecting your information has strengths and weaknesses, both in usability and in the security of the protection system itself, which is why the search for more advanced authentication methods is not over. This project carried out a comprehensive investigation into a variety of authentication methods, considering both those of general applications and those specific to smartphones. In particular, pattern lock authentication was analyzed, identifying an inherent flaw in this system, the vulnerability to over-the-shoulder/shoulder-surfing attacks.

Finally, a prototype was built, PatternLock Scrambler, a method which aims to minimize the effectiveness of over-the-shoulder attacks. The prototype uses transformation mechanisms to input a pattern design by a normal user and apply transformative operations. This is so that an attacker spying on the victim does not realize that the inserted pattern is not the valid pattern for all cases.

To assert the validity of this solution, a questionaire was supplied to end-users that allowed for the conclusion that PatternLock Scrambler works when used in applications that require a high-level security and are not used often.

ÍNDICE

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

2FA	Two-Factor Authentication.
ABC	A lista de acrónimos deve ficar ordenada alfabeticamente.
ADSL	Assimetric Digital Subscriber Line.
AOSP	Android Open Source Project.
ASCII	American Standard Code for Information Interchange.
BIOS	Basic Input/Output System.
bit	Digito binário.
Byte	Unidade de informação digital composta por oito bits.
CA	Certificate Authority.
CODEC	COmpression/DECompression.
CPU	Central Processing Unit.
DHCP	Dynamic Host Configuration Protocol.
DLL	Dynamic Link Library.
DNS	Domain Name System.
DoS	Denial of Service.
FIDO2	Fast Identity Online 2.
FTP	File Transfer Protocol.
HTTP	Hypertext Transfer Protocol.

IP	Internet Protocol.
ISP	Internet Service Provider.
JVM	Java Virtual Machine.
MFA	Multi-Factor Authentication.
OAuth	Open Authorization.
OTP	One Time Password.
OTS	Over-The-Shoulder.
SHA	Secure Hash Algorithm.
SO	Sistema Operativo.
SYA	Something You Are.
SYH	Something You Have.
SYK	Something You Know.
TCP	Transmission Control Protocol.
XML	eXtensible Markup Language.

1

INTRODUÇÃO

A utilização generalizada de dispositivos móveis tais como *smartphones* e *tablets* levou a uma necessidade crescente de métodos de autenticação seguros e convenientes. O Android, o sistema operativo móvel mais popular [1], [2], fornece uma gama de métodos de autenticação para proteger o dispositivo e os dados sensíveis armazenados no mesmo. No entanto, a eficácia e usabilidade destes métodos de autenticação ainda são um tópico de pesquisa e desenvolvimento ativo [3]–[6].

O objetivo deste projeto foi investigar o estado atual da autenticação do Android e identificar áreas a melhorar. Esta tese fornece uma revisão abrangente da literatura existente sobre autenticação em sistemas Android, incluindo os tipos de métodos de autenticação, a eficácia e usabilidade dos diferentes métodos, e as lacunas da investigação. A tese apresenta também o desenvolvimento de uma nova solução para autenticação que aborda as limitações de métodos existentes e fornece uma maior segurança, denominada como *PatternLock Scrambler*.

1.1 ÂMBITO

Este projeto foi realizado no âmbito do mestrado em Cibersegurança e Informática Forense, no seguimento da cadeira Projeto de Segurança, onde se estudou o tema da autenticação *passwordless* como uma defesa aumentada no mundo da Cibersegurança.

A autenticação é uma parte essencial da cibersegurança, logo, um projeto que explore esta área é relevante não só no âmbito de dispositivos móveis mas também da sociedade em geral, visto que a cibersegurança é uma preocupação global.

1.2 CONTEXTO

Nos sistemas informáticos o método de autenticação por omissão, normalmente, é a utilização de uma combinação de utilizador com palavra-passe, ou seja, uma combinação de autenticação baseado no que o utilizador tem (Something You Have

(SYH)) e no que sabe (Something You Know (SYK)) [7], sendo esta a base para distinguir controlo de acesso e autorização entre utilizadores dos sistemas.

Com milhão e meio de *smartphones* vendidos em 2021 [8], uma estimativa de 16,8 mil milhões [9] de *smartphones* existentes no ano de 2023 e quase 7 mil milhões de subscrições em 2023 [9], é garantido haver pessoas que utilizam os seus *smartphones* para guardar informação pessoal e privada, como tal, a segurança dos seus dispositivos móveis é essencial. O primeiro ecrã que a maioria dos utilizadores encontra no seu dispositivo [10] é o ecrã de bloqueio, então é fundamental proteger o ecrã de bloqueio, por ser a primeira linha de defesa contra o acesso não autorizado às informações pessoais e sensíveis dos dispositivos móveis.

Como autenticação é a primeira camada de proteção, o aumento de utilizadores e da complexidade dos mesmos sistemas também causa um aumento de ataques a todas as vertentes dessa camada. Isto significa haver necessidade de sistemas de segurança que protegem das vulnerabilidades presentes na autenticação por palavra-passe. Sistemas de autenticação *passwordless* podem ser a solução que responde a esta necessidade.

Já há algum tempo [5] que equipas de tecnologias de informação tentam substituir a autenticação por palavra-passe por algo que solucione as falhas da palavra-passe sem perder o fator da conveniência associado ao método antigo, mas a necessidade de manter o método conveniente, garantir que autenticação, se for por terceiros (como aplicações de One Time Password (OTP) ou de *push notifications*) não têm *backdoors*, certificar ser resiliente, que não têm falhas inerentes e que tem suporte para identificação de utilizadores independentemente do meio.

Mesmo assim, métodos sem palavra-passe continuam a ter falhas inerentes que têm de ser resolvidas, logo com este projeto espera-se desenvolver um método sem palavra-passe que evita sofrer dos mesmos problemas e, ao mesmo tempo, usufruir, até um certo ponto, das vantagens da palavra-passe.

1.3 DECLARAÇÃO DO PROBLEMA

Os ataques *shoulder-surfing* ou Over-The-Shoulder (OTS) são um risco de segurança comum em que um atacante observa física e secretamente a vítima a introduzir informações pessoais e sensíveis tais como, palavras-passe, PINs, padrões de desbloqueio ou outros dados privados. Este tipo de ataque pode ocorrer numa variedade

de locais, como caixas multibanco, transportes públicos, cafés, locais de trabalho, entre vários outros [11].

Os ataques OTS são problemáticos para a autenticação gráfica, principalmente em *smartphones*[12], logo, como a utilização generalizada de *smartphones*, *tablets* e outros dispositivos portáteis permitem o acesso rápido a informações sensíveis, ataques a dispositivos com este tipo de autenticação podem causar roubo ou perda de dados sensíveis. Além disso, a utilização generalizada de redes Wi-Fi públicas, bem como a tendência crescente para trabalhar à distância, coloca os utilizadores em maior risco de se tornarem vítimas destes ataques.

Como a execução bem sucedida de um ataque OTS pode resultar em consequências graves, como usurpo de identidade, fraude financeira ou acesso ilegal a contas pessoais e profissionais, estes incidentes podem resultar perdas financeiras, deterioração de saúde mental e danos à reputação[13]. Então, é fundamental desenvolver estratégias eficazes, defesa contra ataques OTS e preservar a privacidade e a segurança das informações sensíveis das pessoas, à medida que a tecnologia continua a avançar e a tornar-se uma parte essencial do nosso quotidiano.

Existem medidas anti-OTS como ecrãs de privacidade, mecanismos de autenticação biométrica e abordagens baseadas no comportamento, mas estas podem não proporcionar uma proteção eficaz contra atacantes dedicados ou podem ser impraticáveis para uma implementação geral. Além disso, estas soluções necessitam frequentemente de um compromisso entre a segurança e a experiência do utilizador, exigindo o desenvolvimento de técnicas novas e fáceis de utilizar para combater ameaças OTS.

Um artigo apresentado por Ye et al. [11] explora exatamente estas fraquezas e traz à luz os problemas que se apresentam. O presente projeto procura mitigar a fraquezas apresentadas nesse artigo.

1.4 OBJETIVOS DO PROJETO

Este projeto tem o objetivo de se realizar uma revisão da história da cibersegurança relativamente à autenticação até à introdução nos métodos contemporâneos de autenticação, uma investigação dos trabalhos relacionados de interesse, e uma exploração das fraquezas da autenticação por palavra-passe e como estas poderão ser corrigidas com as alternativas disponíveis. Seguidamente foi feita uma investiga-

ção pormenorizada sobre os mecanismos encontrados com intento de perceber as fraquezas de cada e como deve ser possível eliminar as mesmas.

Este projeto teve em vista reduzir o perigo destes ataques tentando não reduzir drasticamente a experiência do utilizador, e como tal, explora novas metodologias e estratégias para prevenir esses ataques.

O projeto, finalmente, visou apresentar um novo método desenvolvido que tem em vista mitigar o perigo de ataques de *shoulder-surfing*, o *PatternLock Scrambler*.

1.5 ESTRUTURA DO RELATÓRIO

Este relatório começa pela presente introdução que apresenta uma revisão breve da área da segurança de informação relacionada com este projeto, explica o âmbito em que este projeto foi realizado, define o contexto em que o mesmo se encontra, estabelece o problema encontrado a ser resolvido e assenta os objetivos do projeto.

Seguidamente, no Capítulo 2 foi realizada uma revisão da área de segurança relevante, são apresentadas as tecnologias utilizadas no desenvolvimento prático, realizada uma visão geral da autenticação no ambiente Android onde se apresentam métodos como PINs e palavras-chave, métodos biométricos, padrões e *smartlocks*. Depois foram analisados os métodos relacionados com smartphones, mas que não são propriamente utilizados no desbloqueio e na utilização dos mesmos, como One Time Password (OTP) e Multi-Factor Authentication (MFA) (mecanismos regularmente instalados como Apps em *smartphones* para autenticar outros serviços). Adicionalmente foi explorado o mecanismo FIDO2, um mecanismo que utiliza a criação de chaves privadas e partilha de chaves públicas por meio de *challenges* e assinaturas[14]. Finalmente, neste capítulo, foi realizada uma revisão literária de artigos e publicações científicas da área, principalmente sobre um artigo que apresentou fraquezas num método de autenticação que o presente projeto teve em vista resolver.

No capítulo seguinte Capítulo 3, está descrita a arquitetura global do sistema construído durante a realização deste projeto, é explicado o conceito geral do programa desenvolvido, como foi planeado funcionar previamente à sua construção e apresentados mockups das páginas presentes na aplicação.

O Capítulo 4 é o capítulo do desenvolvimento da prova de conceito construída no decorrer deste projeto. Aqui estão apresentados excertos de código, amostras

da interface de utilizador e a implementação seguida, tal como um exemplo da utilização da aplicação que funciona como prova de conceito.

Capítulo 5 é o capítulo onde se recolheram as respostas a um questionário fornecido a utilizadores que testaram a aplicação. As respostas foram normalizadas e analisadas para extrair conclusões sobre a utilização da aplicação final do ponto de vista de um utilizador normal no seu quotidiano.

Por fim, o capítulo da conclusão é o Capítulo 6, onde são apresentadas todas as conclusões retiradas no desenvolvimento deste projeto e explicitadas as fraquezas do mesmo, tal como os pontos que se podem aprofundar em desenvolvimento.

2

REVISÃO DE LITERATURA

Os *smartphones* modernos são uma peça essencial do quotidiano do cidadão do século XXI [15]–[17]. Com a ligação do mundo pela internet e conveniência que um *smartphone* traz, é normal que utilizadores guardem informação sensível nos seus telemóveis. Como tal, é indispensável que a segurança desses dispositivos esteja sempre a evoluir para impedir que maus atores roubem dados importantes. Não se pode deixar à sorte a segurança da informação privada, mas também, há um limite de segurança que, para o utilizador comum, é demasiado inconveniente para utilizar no dia a dia. Então, há um equilíbrio a manter entre conveniência, funcionalidade e segurança, procurando-se com cada mecanismo maximizar estas três, sempre uma ao custo das outras duas [18].

2.1 TECNOLOGIAS RELEVANTES

O sistema operativo Android é um sistema operativo *open-source* com base no *kernel* Linux para dispositivos móveis, como *smartphones* e *tablets*, mas que agora engloba mais tipos de dispositivos. Foi concebido principalmente para dispositivos com ecrã tátil. O Android OS é conhecido pela sua natureza de código aberto, que permite aos fabricantes e programadores personalizar e modificar o sistema operativo consoante os seus requisitos de hardware e software. Android é desenvolvido pela *Open Handset Alliance* e Google [2].

Kotlin é o nome de uma linguagem de programação baseada na Java Virtual Machine (JVM). A linguagem que Google recomenda que seja usada inicialmente para criar aplicações para Android é Kotlin, passando apenas para outras linguagens se Kotlin não for aplicável. A empresa mãe já anunciou que todas as aplicações novas são criadas com uma mentalidade *Kotlin first* (Kotlin primeiro) pois assegura que Kotlin é uma língua mais adequada para novo software comparada com a sua base Java [19], [20].

Conforme a Google, o desenvolvimento em Kotlin torna o código mais acessível, reduz a quantidade de erros comuns (pois engloba funções estáveis em Java), tem

interoperabilidade com Java, tem um crescimento estável e seguro, é fácil de aprender e a sua comunidade cresce exponencialmente.

2.2 VISÃO GERAL DA AUTENTICAÇÃO EM ANDROID

Este capítulo descreve os conceitos necessários e essenciais ao desenvolvimento deste trabalho, enumerando métodos e ferramentas para efetuar uma autenticação válida e segura tendo como base o sistema operativo móvel Android. A autenticação em dispositivos móveis pode ser dividida em abordagens implícitas [13], [21] e explícitas [22]. Além disso, existem abordagens mistas [23] que adicionam camadas de segurança implícitas a um desafio de autenticação explícita. Mecanismos de autenticação implícita analisam períodos específicos de pistas de comportamento como dados de sensores e padrões de uso para estabelecer uma autenticação contínua, portanto, reduzir a carga de trabalho de autenticação. Exemplos incluem a análise de padrões de marcha [24], comportamento de digitação [25], acesso ao sistema de ficheiros [26], ou uma combinação de fatores [27]. Devido a atrasos notáveis, muitos deles não são adequados para mecanismos de bloqueio direto de ecrã.[28]

A autenticação em android segue as mesmas três camadas de autenticação que outros métodos utilizam: Something You Know (SYK), Something You Have (SYH) e Something You Are (SYA). O uso intercalado das três camadas de autenticação (SYK, SYA e SYH) é uma prática comum em sistemas de segurança mais avançados, incluindo algumas aplicações e dispositivos Android que requerem maior nível de proteção [29], [30].

A utilização intercalada destas camadas na autenticação móvel desempenha um papel crucial no reforço da segurança global das aplicações Android. Ao combinar vários fatores de autenticação, os programadores podem estabelecer uma defesa robusta e multifacetada contra o acesso não autorizado e potenciais ameaças cibernéticas. Será aprofundada cada camada e explorada a sua importância no cenário de autenticação móvel na Subseção 2.3.1.

2.2.1 *Palavra-passe e PIN*

Uma combinação definida pelo utilizador de letras, números e/ou símbolos que devem ser introduzidos para aceder ao dispositivo. Este tipo de autenticação requer que o utilizador se lembre do que ele sabe. Há duas partes neste método. Primeiro,

o utilizador introduz o nome de utilizador e segundo, a palavra-passe. A senha é a combinação secreta de palavras e números que o utilizador conhece [30]–[32].

Sendo o método mais comum e mais utilizado, as vantagens já são mais que conhecidas pelo mundo e a única razão por ainda ser utilizado é:

- Fácil recordação para o utilizador
- Mais fácil implementação pelo programador
- Familiaridade do utilizador

Os problemas da utilização deste método é que: utilizadores tendem a armazenar as suas palavras-passe em texto simples e as suas informações privada nos *smartphones* para realizar operações eficientemente e sem complicações; ataques por engenharia social conseguem eliminar o fator de encriptação de uma palavra-passe; utilizadores podem reutilizar palavras-passe, indiferente se são fracas ou fortes, facilitando a exploração de um *leak* de palavras-passe de um *website* para utilização noutro (*password stuffing*); é possível detetar no ecrã marcas do óleo da pele dos utilizadores e assim deduzir padrões ou PINs; ataques como *shoulder surfing*, que usa técnicas de observação direta como o olhar sobre o ombro de um utilizador para ver os códigos; até palavras-passe fortes são vulneráveis a ataques [30], [32]–[35].

2.2.2 *Autenticação biométrica*

Este tipo inclui vários métodos que usam as características biológicas únicas do utilizador para autenticar a sua identidade, como, por exemplo, no caso físico:

- Reconhecimento de impressões digitais: Digitalização e correspondência da impressão digital do utilizador com a que está armazenada no dispositivo.
- Reconhecimento facial: Utilização da câmara frontal do dispositivo para digitalizar e combinar a cara do utilizador com a que está armazenada no dispositivo.
- Reconhecimento da íris: Digitalização e correspondência do padrão único da íris do utilizador com a que está armazenada no dispositivo.
- Reconhecimento de voz: Analisar e comparar a voz do utilizador com a voz armazenada no dispositivo.
- Reconhecimento da palma: Digitalização e correspondência da palma da mão do utilizador com a que está armazenada no dispositivo

E no caso comportamental:

- *Keystroke*: Isto envolve a análise do ritmo e padrão da digitação de um utilizador para autenticar a sua identidade. Por exemplo, o tempo entre as teclas e a duração de cada tecla pode ser usado para criar um perfil único para cada utilizador [31].
- *Touchscreen*: O comportamento do *touchscreen* refere-se à forma como um indivíduo interage com um dispositivo *touch*, incluindo a pressão, a velocidade, a duração e o padrão dos seus movimentos tácteis [36].
- *Signature*: Comparação da assinatura do utilizador com assinaturas prévias [37].
- *Behavior Profiling*: Criação de um perfil comportamental do utilizador real durante a atividade normal para comparação com outro utilizador[30].
- *Handwaving*: Desbloqueio do *smartphone* ao abanar o mesmo [38] ou a mão do utilizador [39].

As vantagens que a autenticação biométrica oferece são que características inerentes biométricas não podem ser perdidas, esquecidas, adivinhadas, roubadas, partilhadas, etc; são fáceis verificar se uma pessoa tem várias identidades; e a implementação de autenticação com base no comportamento é geralmente económica devido a não se necessitar de hardware especializado e por serem implementações leves [37]. No entanto, em alguns casos, pode ser emitida uma autenticação falsa, usualmente por pessoas semelhantes; não é substituível nem secreto; se os dados biométricos de uma pessoa são roubados, não é possível substituí-los. Os métodos de autenticação biométrica também sofrem o problema de que a taxa de falsos negativos e falsos positivos são mais elevadas comparados aos outros métodos. Por exemplo, em [33] os autores analisaram um método de autenticação chamado *Typing Authentication and Protection* com uma taxa de falsos positivos de 8.93% [30], [32], [33], [35], [36], [38], [40]–[43].

Google publicou uma lista de instruções e informações a considerar quando se utiliza a autenticação biométrica. Estas instruções são sobre a linha de modelos *smartphone* Pixel, no entanto, aplicam-se a vários modelos [44].

INSTRUÇÕES DA GOOGLE

- Os modelos faciais podem ser considerados dados biométricos em algumas jurisdições.

- Quando olha para o seu telefone, pode desbloqueá-lo mesmo quando não tem intenção de o fazer.
- O desbloqueio facial pode ser menos seguro do que um PIN, padrão ou palavra-passe forte.
- O seu telefone pode ser desbloqueado por alguém que se pareça muito consigo, como um irmão idêntico.
- O seu telefone também pode ser desbloqueado por outra pessoa, se for segurado até à sua cara. Mantenha o seu telefone num local seguro, como o seu bolso da frente ou a sua mala de mão. Para se preparar para situações inseguras, aprenda a ligar o bloqueio.
- Quando não houver luz suficiente ou quando tiver uma cobertura facial ou óculos de sol, o Desbloqueio Facial pode não funcionar.

2.2.3 *Smart Lock*

Smart Lock permite aos utilizadores desbloquear automaticamente o seu dispositivo quando certas condições são cumpridas, tais como quando o dispositivo está num local de confiança ou quando deteta um dispositivo de confiança nas proximidades. Consoante a Google, em alguns aparelhos, a deteção no seu bolso aprende o padrão de marcha do utilizador através dos sensores do *smartphone*, nomeadamente o velocímetro. Pelos sensores, o *SmartLock* consegue verificar se o utilizador que transporta o dispositivo é verdadeiro, pois um atacante poderá ter um padrão e velocidade de marcha distinto do utilizador real [31], [45].

- *On-body Detection*: usa vários padrões biométricos comportamentais (isto é, padrões de marcha e movimento do corpo) para manter o telefone desbloqueado enquanto estiver em movimento. Pode também bloquear automaticamente o telefone se nenhum movimento for detetado [31], [46].
- *Trusted Places*: utiliza sinais GPS e Wi-Fi para desbloquear o telefone em locais específicos (por exemplo, a casa de um utilizador) automaticamente. Também pode bloquear automaticamente o telefone quando o dispositivo deixa o local de confiança [46].
- *Trusted Devices*: usa sinal *Bluetooth* para bloquear e desbloquear o telefone. Os utilizadores podem designar dispositivos Bluetooth já emparelhados como dispositivos de confiança, permitindo-lhes desbloquear o telefone automaticamente quando é estabelecida uma ligação. Isto pode também bloquear

automaticamente o telefone quando este perde a hiperligação com todos os dispositivos de confiança[1] [46], [47].

O que o *SmartLock* traz à mesa é a velocidade de desbloqueio, a conveniência para o uso diário em locais seguros, e o desbloqueio sem necessitar interação do utilizador. Mas os *trusted places* são uma estimativa de localização que depende do GPS e outros sensores, ou seja, como o GPS não é sempre preciso/certeiro, permite que no roubo do dispositivo este mesmo se desbloqueie num lugar não seguro desde que seja perto do *trusted place* porque a sua localização de confiança pode ir para além das paredes da sua casa ou local personalizado e pode manter o seu telefone desbloqueado num raio de até 80 metros, o *SmartLock* também continua a necessitar de interação fora das condições normais, por exemplo, no autocarro e num avião os sensores não conseguem deduzir que a autenticação é legítima e os sinais de localização podem ser copiados ou manipulados. Adicionalmente, alguém com acesso a equipamento especializado pode desbloquear o seu telefone [46], [48].

Google também publicou as instruções e precauções a tomar quando se utiliza *smart locks* em Android [48].

2.2.4 *Desenho de padrão*

A autenticação através do desenho de um padrão é um método que requer que o utilizador defina um padrão gráfico nas definições do *smartphone* para que sempre que quiser desbloquear o dispositivo o tenha de redesenhar, tal como se define um PIN que se reintroduz ao desbloquear. O desenho de um padrão específico no ecrã táctil do dispositivo para o desbloquear é um método bastante utilizado, em que, segundo [28], 8.5% da população por eles testada, usa essa técnica.

Como a autenticação por padrão é de fácil utilização, principalmente quanto mais usada, pode ser mais rápida do que uma palavra-passe; também é um conceito familiar, permite uma personalização em alguns dispositivos, que disponibilizam definições para alterar o número de pontos do padrão; e não requerem hardware específico como sensores digitais debaixo do ecrã [28], [30], [40], [49].

O problema é que a autenticação de padrões é vulnerável a ameaças à segurança, tais como *shoulder surfing* e ataques de manchas (marcas de óleo da pela), onde alguém pode observar ou deduzir o padrão a partir das manchas no ecrã. Tal como em PINs e senhas, os utilizadores podem escolher padrões simples e previsíveis,

1 Engloba smart cards

tais como um simples deslize, que pode ser facilmente adivinhado ou deduzido por um atacante, por contrário, a autenticação padrão é limitada em complexidade em comparação com outros métodos de autenticação, tais como palavras-passe ou biometria. Noutra vertente, a autenticação do padrão requer uma entrada precisa no ecrã táctil, o que pode ser um desafio para os utilizadores com dedos maiores ou incapacidades. Finalmente a dificuldade do padrão depende do comprimento do mesmo, tal como nas senhas, o que pode tornar o padrão inconveniente de utilizar [11], [30], [40], [49].

O artigo mencionado no Capítulo 1 foi investigado e aprofunda nas fraquezas deste método, do qual este projeto se baseia parcialmente ao tentar mitigar as fraquezas aí apresentadas [11].

Este artigo [11], apresenta um ataque ao bloqueio de padrões do Android que se baseia em vídeo, este ataque pode inferir o padrão através do rastreio dos movimentos das pontas dos dedos, sem exigir a captura de qualquer conteúdo apresentado no ecrã. A abordagem identifica com precisão um pequeno número de padrões candidatos a serem testados e consegue quebrar mais de 95% dos padrões em cinco tentativas antes de o dispositivo ser automaticamente bloqueado. O estudo concluiu que os padrões complexos não oferecem uma proteção mais forte do que os simples e apela a uma revisão dos riscos associados à utilização do bloqueio de padrões do Android para proteger informações sensíveis.

O ataque apresentado começa com uma gravação externa por um atacante que apanha a vítima a desbloquear o seu telemóvel despercebida do atacante. Cerca de 2 a 3 metros de distância entre o atacante e a vítima é suficiente para a gravação funcionar. Assim que o atacante grava um desbloqueio do ecrã pode passar o vídeo para o sistema que, em pouco tempo, devolve até 5 padrões possíveis a experimentar.

O sistema funciona tão bem que até consegue resolver os padrões mais complexos 80% das vezes à primeira tentativa, enquanto os mais simples só tem entre 50 a 60% de probabilidade de resolver à primeira. A discrepância existe porque em padrões mais complexos o sistema tem mais referências de movimento e então consegue deduzir e descartar os padrões que não caberiam no movimento.

O sucesso do ataque depende de três fatores:

1. conhecimento da grelha de padrões;
2. uma gravação de vídeo de qualidade decente que permita ao algoritmo seguir o movimento da ponta dos dedos;

3. identificar com sucesso um segmento de vídeo que capte todo o processo de desenho do padrão.

Os autores discutem contramedidas possíveis ao sistema que desenvolveram. Estas contam com técnicas de aleatorização, como as imagens aleatórias, que dizem poder ser uma solução para o primeiro fator, mas descartam tal por poder causar demasiada inconveniência ao utilizador. Para o fator 2, hipotetisam a utilização de um sistema como [50], que utiliza recodificação de vídeo no canal ecrã-camara para diminuir a qualidade de imagem, e assim impedir que um atacante grave a autenticação. O terceiro fator, dizem os autores, é algo que pode ser limitado ao misturar o ato de desbloquear no meio de outras ações para que o atacante nunca saiba quando é que o desbloqueio comece, mas perante essa complicação ao atacante está uma também para o utilizador que terá de desbloquear o ecrã dessa maneira sempre que quer utilizar o ecrã.

No fim de tudo, os autores concluem que aumentar a complexidade do padrão é uma ilusão de segurança que até acaba por trair o utilizador e que o bloqueio por padrão do Android é vulnerável a ataques baseados em vídeo.

2.3 CONTEXTO RELACIONADO

Os métodos de autenticação dos *smartphones* evoluíram ao longo do tempo, desde as palavras-passe tradicionais até às tecnologias biométricas avançadas, oferecendo aos utilizadores formas seguras e convenientes de aceder aos seus dispositivos. Tipicamente esta autenticação ocorre no início da interação com o dispositivo em causa, mas, com a evolução dos métodos, a maneira de que é feita às vezes é bastante distinta do que mais estamos familiarizados. Como tal é necessário demonstrar os mesmos e como funcionam.

Regra geral, autenticação pelo menos um dos 3 tipos gerais, cada um baseado numa característica diferente do utilizador [51].

2.3.1 *Multi-Factor Authentication*

À medida que as organizações digitalizam as operações e assumem uma maior responsabilidade pelo armazenamento de dados dos clientes, os riscos e a necessidade de segurança aumentam. Uma vez que os atacantes exploram há muito os dados

de início de sessão dos utilizadores para aceder a sistemas críticos, a verificação da identidade dos utilizadores tornou-se essencial [52].

A autenticação baseada apenas em nomes de utilizador e palavras-passe não é fiável e é difícil de manusear, uma vez que os utilizadores podem ter dificuldade em armazenar, recordar e gerir essas palavras-passe em várias contas e muitos reutilizam palavras-passe em vários serviços e criam palavras-passe pouco complexas. As palavras-passe também oferecem uma segurança fraca devido à facilidade de as adquirir por meio de pirataria informática, *phishing* e *malware*.

Multi-Factor Authentication (MFA) é uma funcionalidade de segurança que necessita de 2 ou mais fatores de autenticação para validar a identidade do utilizador, como a combinação de uma palavra-passe com a posse de um *smartphone* validado, do acesso à conta de *email* associada a onde o utilizador se valida, ou da impressão digital. Métodos comuns de MFA contam com a combinação de palavras-passe/utilizador com um código de uso único enviado para o email do utilizador ou gerado no *smartphone* do utilizador, necessário inserir na página de autenticação; o envio de uma notificação para o dispositivo do utilizar que precisa de ser pressionada para validar a entrada, e o envio de um email com um *hyperlink* que o utilizador precisa de seguir, entre outros [4], [52], [53]. As caixas multibanco, por exemplo, utilizam MFA quando requerem que o utilizador insira o seu cartão e o PIN respetivo na máquina para levantar dinheiro.

A MFA requer meios de verificação que os utilizadores não autorizados não terão. Uma vez que as palavras-passe são insuficientes para verificar a identidade, a MFA requer várias provas para verificar a identidade. A variante mais comum da MFA é a autenticação de dois fatores Two-Factor Authentication (2FA). A teoria é que, mesmo que os atacantes consigam fazer-se passar por um utilizador com uma prova, não conseguirão fornecer duas ou mais.

A autenticação multifator adequada utiliza fatores de pelo menos duas categorias diferentes. Utilizar dois da mesma categoria não cumpre o objetivo da MFA. Apesar da utilização generalizada da combinação palavra-passe/pergunta de segurança, ambos os fatores são da categoria conhecimento e não se qualificam como MFA. Uma palavra-passe e um código de acesso temporário qualificam-se porque o código de acesso é um fator de posse, assim sendo SYK (credenciais) e SYH (fator adicional), verificando a propriedade de uma conta de correio eletrónico ou dispositivo móvel específico [54].

2.3.2 *One Time Password*

One Time Password (OTP), ao contrário do que o nome indica, não sofrem dos mesmos problemas que as *passwords* usuais, devido ao limite temporal destas senhas, as OTPs são utilizadas como uma medida de autenticação que valida o utilizador por outros meios. De acordo com NIST [52] um dispositivo OTP multifator gera OTPs para o uso em autenticação após ativação por meio de um fator adicional de autenticação. Isto inclui tanto geradores OTPs em hardware como aplicações/software geradores em dispositivos como *smartphones*. O segundo fator de autenticação pode ser obtido via um teclado, uma parte leitora de biométricas ou outras interfaces computacionais. A OTP é apresentada no ecrã onde o utilizador irá ler o valor e seguidamente inserir manualmente na entidade verificadora. O dispositivo multifator OTP é SYH, tanto em hardware, como em software, e será ativado por ou SYA ou SYK.

Os dispositivos OTP devem seguir regras estritas para gerar OTPs, por exemplo, deve conter 2 valores persistentes: uma chave simétrica que persiste no dispositivo na vida toda, e um *nonce* [2] que é alterado sempre que um OTP é usado, ou que é baseado numa *seed* de tempo. A chave simétrica e o algoritmo de um OTP deve oferecer uma força de pelo menos 112 *bits*, o *nonce* tem de ser de tamanho suficiente para gerar sempre um código distinto para cada operação, e o dispositivo em si deve dissuadir e tentar prevenir a clonagem da chave secreta para vários dispositivos. O *nonce* é baseado numa *seed* temporal, então é obrigatório alterar pelo menos a cada 2 minutos [52].

Para OTP funcionar corretamente deve haver um verificador OTP do lado do sistema a ser acedido. O verificador OTP segue as mesmas regras que o gerador OTP, no entanto, não precisa de um segundo fator para ser usado, logo deve ser fortemente protegido contra ataques. Este verificador tem uma cópia da chave simétrica no dispositivo OTP. O verificador funciona a partir da chave que guarda. Tanto o gerador como o verificador, ao usar a chave simétrica, vão gerar um *nonce* idêntico, assim, quando o utilizador envia o OTP para o sistema em que se tenta autenticar, o verificador certifica que o *nonce* gerado por si é igual ao recebido vindo do utilizador.

2 *Nonce*:

> *A nonce is a random or semi-random number that is generated for a specific use. It is related to cryptographic communication and information technology (IT). The term stands for "number used once"or "number once"and is commonly referred to as a cryptographic nonce* [55].

Se o *output* do autenticador tiver menos de 64 *bits* de entropia, o verificador necessita de implementar um mecanismo limitador de tentativas de autenticação falhadas.

Existem dispositivos físicos e software gerador de OTP, logo, há naturalmente dispositivos mais e menos famosos. Yubico YubiKey são das mais conhecidas chaves físicas geradoras de OTPs. Estas chaves são inseridas no dispositivo em que procura autenticar-se. Google Authenticator, das apps OTP mais populares, é uma aplicação OTP criada pela Google que armazena chaves simétricas de OTPs para o uso mais facilitado e conveniente (pois tem o código no próprio dispositivo).

2.3.3 *Open Authorization*

> *OAuth 2.0 is the industry-standard protocol for authorization. OAuth 2.0 focuses on client developer simplicity while providing specific authorization flows for web applications, desktop applications, mobile phones, and living room devices. This specification and its extensions are being developed within the IETF OAuth Working Group* [56].

No modelo tradicional de autenticação cliente-servidor, o cliente solicita um recurso de acesso restrito (recurso protegido) no servidor, autenticando-se no servidor com as credenciais do proprietário do recurso. Para fornecer a aplicações de terceiros acesso a recursos restritos, o proprietário do recurso partilha as suas credenciais com o terceiro. Isto cria vários problemas e limitações:

- As aplicações de terceiros são obrigadas a armazenar as credenciais do proprietário do recurso para utilização futura, normalmente uma palavra-passe em texto claro.
- Os servidores são obrigados a suportar a autenticação por palavra-passe, apesar das fragilidades de segurança inerentes às palavras-passe.
- As aplicações de terceiros obtêm um acesso demasiado amplo aos recursos protegidos do proprietário do recurso, deixando os proprietários do recurso sem qualquer capacidade de restringir a duração ou o acesso a um subconjunto limitado de recursos.
- Os proprietários de recursos não podem revogar o acesso a um terceiro individual terceiros sem revogar o acesso a todos os terceiros, e devem fazê-lo alterar a palavra-passe do terceiro.

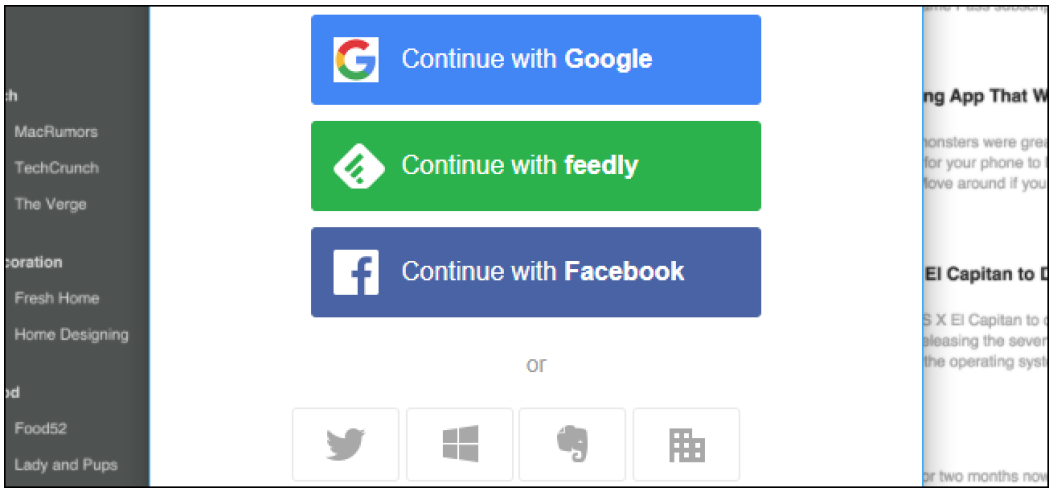


Figura 1: Exemplo da interface visual do ponto de vista do utilizador para iniciar sessão por OAUTH (Fonte: Geeks4Geeks.org)

- O comprometimento de qualquer aplicação de terceiros resulta no comprometimento da palavra-passe do utilizador final e todos os dados protegidos por essa palavra-passe.

Open Authorization (OAuth) é um método de autenticação que funciona através da partilha de *tokens* autenticados entre programas/sistemas/*websites*. Um exemplo de autenticação através de OAuth é apresentado na Figura 1 que foi retirada de um *website* onde se pode ver que o utilizador se pode autenticar via *Facebook*, *Google* e *feedly*. A estrutura de autorização OAuth 2.0 permite que uma aplicação de terceiros obtenha acesso limitado a um serviço Hypertext Transfer Protocol (HTTP), quer em nome do proprietário de um recurso, orquestrando uma interação de aprovação entre o proprietário do recurso e o serviço HTTP, ou permitindo que a aplicação de terceiros obtenha acesso em seu nome, através dos seguintes passos [57]:

1. O protocolo inicia com um cliente a pedir autorização ao proprietário de um recurso, tanto diretamente, como por meio de um intermediário que funciona como servidor de autorização.
2. O cliente recebe a credencial que representa a autorização do proprietário. Esta credencial pode ser um de 4 tipos (código de autorização, implícito, senha do proprietário do recurso, credenciais do cliente).
3. O cliente requisita um *token* de acesso ao servidor de autorização ao se autenticar no mesmo com a credencial de autorização recebida.
4. O servidor autentica e valida a credencial e emite um token de acesso caso válida

5. O cliente pede acesso ao servidor do recurso autenticado-se com o token de acesso recebido
6. O servidor do recurso disponibiliza o recurso ao cliente conforme o nível de acesso, caso este seja válido

2.3.4 *Certificados*

Um certificado é um documento eletrónico utilizado para identificar um indivíduo, um servidor, uma empresa ou outra entidade e para associar essa identidade a uma chave pública [58]. Os certificados utilizam criptografia de chave pública para resolver o problema da falsificação de identidade.

Os certificados funcionam como formas de identificação. As autoridades de certificação ou Certificate Authority (CA) são entidades que validam identidades e emitem certificados. Os clientes e servidores utilizam certificados emitidos pela CA para determinar os outros certificados em que podem confiar. Tal como os métodos de validação de outras formas de identificação podem variar em função de quem emite a identificação e da finalidade para que está a ser utilizada, os métodos utilizados para validar uma identidade podem variar em função das políticas de uma determinada CA. Em geral, antes de emitir um certificado, a CA deve utilizar os seus procedimentos de verificação publicados para esse tipo de certificado para garantir que uma entidade que solicita um certificado é, de facto, quem afirma ser.

O certificado emitido pela CA vincula uma determinada chave pública ao nome da entidade que o certificado identifica; por exemplo, o nome de um funcionário ou de um servidor. Os certificados ajudam a evitar a utilização de chaves públicas falsas para fins de falsificação de identidade. Apenas a chave pública certificada funcionará com a chave privada correspondente, que é propriedade da entidade certificada.

Para além de uma chave pública, um certificado também inclui o nome da entidade que identifica, uma data de validade, o nome da CA que emitiu o certificado, um número de série e outras informações. Mais importante ainda, um certificado inclui sempre a assinatura digital da CA emissora. A assinatura digital da CA permite que o certificado funcione como uma carta de apresentação para os utilizadores que conhecem e confiam na CA, mas não conhecem a entidade identificada pelo certificado.

A autenticação é o processo de confirmação de uma identidade. No contexto da forma como uma rede interage, a autenticação envolve a identificação confiante de uma parte por outra parte. A autenticação numa rede pode assumir muitas formas, e a utilização de certificados é uma forma de suportar essa autenticação.

As interações em rede ocorrem geralmente entre um cliente e um servidor. A autenticação do cliente é a identificação segura de um cliente por um servidor, ou seja, a identificação da pessoa que se presume estar a utilizar o cliente. A autenticação do servidor é a identificação segura de um servidor por um cliente, ou seja, a identificação da organização que se presume ser responsável pelo servidor num determinado endereço de rede. A autenticação baseada em certificados baseia-se no que o utilizador tem, o qual é a chave privada do utilizador, e no que o utilizador sabe, sendo a palavra-passe que protege a chave privada (se a chave não estiver localizada num repositório de chaves seguro). No entanto, ambos os pressupostos só são verdadeiros se pessoal não autorizado não tiver obtido acesso à estação de trabalho ou à palavra-passe do utilizador, se a palavra-passe da base de dados de chaves privadas do cliente tiver sido definida e se o cliente estiver configurado para solicitar a palavra-passe em intervalos razoavelmente frequentes. Embora a autenticação baseada em certificados trate da segurança, não trata de questões relacionadas com o acesso físico a estações de trabalho ou palavras-passe individuais. A criptografia de chave pública apenas verifica se uma chave privada utilizada para assinar algumas informações corresponde à chave pública num certificado. É da responsabilidade do utilizador proteger a segurança física de uma estação de trabalho e manter a palavra-passe da chave privada em segredo.

2.3.5 *Fast Identity Online 2*

Fast Identity Online 2 (FIDO2) é um mecanismo de autenticação que visa eliminar o uso de palavras-passe ao nível do servidor. Esta remoção de palavras-passe *server-side* provem do facto que a FIDO Alliance [14] alega que 80% de todas as *data breaches* tem origem em palavras-passe e que mais de metade de todas as palavras-passe são reutilizadas. FIDO2 elimina o uso de palavras-passe utilizando, por sua vez, um armazenamento de chaves públicas geradas por cada cliente no seu próprio dispositivo: por meio de técnicas de criptografia de chave pública durante o registo de um utilizador num serviço, o cliente FIDO2 no dispositivo do utilizador gera um par de chaves pública e privada; e assim, em vez de ter de comparar um *hash* de senha com o guardado na base de dados, o servidor precisa apenas de enviar um

desafio ao cliente que se aspira autenticar. As respostas a desafios são autenticadas através do uso das chaves privadas como cifras que os assinam. A assinatura do desafio faz com que possa ser utilizada a chave pública que faz par com essa como verificação de autenticidade.

A Figura 2 apresenta o fluxo que *standards* como o FIDO2 seguem aquando uma tentativa de autenticação.

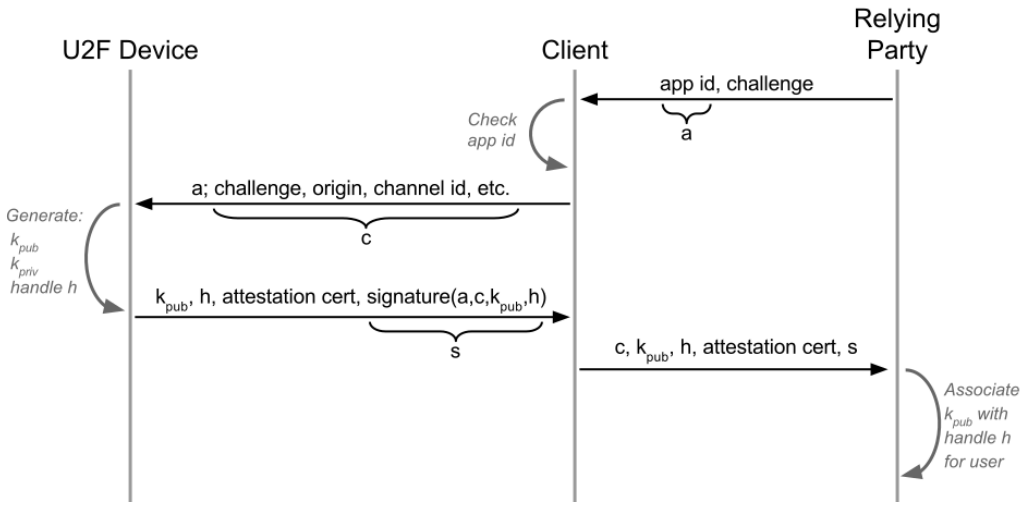


Figura 2: Fluxo de autenticação e registo simplificado através do FIDO2 (Fonte: *Yubico Website*[59])

O projeto final de mestrado de Pastilha [60] aprofunda sobre o FIDO2 de uma maneira mais concreta. Foca-se no mecanismo FIDO2 e como este pode ser implementado na aplicação Android "BitWarden", um gestor de palavras-passe, para o uso na autenticação da mesma. O documento apresenta um resumo dos mecanismos atuais de autenticação, incluindo tanto os relevantes para o problema apresentado como os que fazem apenas parte da área. Sendo o foco o FIDO2, a tese aprofunda bastante este mecanismo, explicando como funciona detalhadamente.

Pastilha aponta para cada mecanismo, apresenta os seus casos de uso, as vantagens que cada um traz e ilumina as falhas inerentes a cada, nomeadamente das palavras-passe:

- Palavra-passe
- Múltiplos fatores
- Baseada em chaves públicas
- Biométricas
- Tokens e Nonces

- Yubikeys

Pastilha criou uma prova de conceito para demonstrar a viabilidade do seu projeto, demonstra a arquitetura geral que planeou usar, disponibiliza um *link* que utilizou como demonstração e os fluxos de interações entre o servidor e utilizadores. Depois da prova de conceito, o autor esclarece o que é o "BitWarden"e como o FIDO2 é incorporado pelo autor.

3

ARQUITETURA

Neste ponto do projeto é sabido que cada mecanismo tem problemas inerentes à sua utilização. Este projeto visou aprofundar o mecanismo de autenticação através do desenho de padrão, baseando-se até um pouco no artigo apresentado por Ye et al. [11] para recolher mais informações sobre as suas fraquezas. Desse artigo registamos que uma fraqueza do desenho de padrão é o facto de se poder gravar um utilizar a desenhar o padrão e assim conseguir-se deduzir qual tem de ser desenhado.

Mas e se o padrão que a vítima tem de desenhar não for sempre igual?

Esta é a solução que o presente projeto apresenta. Ao alterar o padrão necessário para autenticação periodicamente, uma gravação OTS proposta em [11] será mitigada, visto que o software não calcularia a resposta correta para autenticar. Para isso, terão de ser implementados mecanismos que recebem um padrão do utilizador, guardam-no como base de comparação e depois aplicam-se necessidades de transformação aos padrões seguintes para que, quando revertida a transformação, sejam idênticos ao padrão inicial guardado.

3.1 ARQUITETURA DO SISTEMA

O sistema foi desenhado de maneira a utilizar 2 atividades e 3 principais operações. Uma atividade é uma classe fulcral que funciona como camada de apresentação o utilizador interagir, como um ecrã que os utilizadores vêm, sendo por via das atividades que aplicações são inicializadas. A Figura 3 apresenta o primeiro nível da arquitetura do sistema em modelo C4.

Esta subsecção descreve a arquitetura do método PatternLock Scrambler que foi construído no Capítulo 4, este método foi aqui arquitetado como software intermediário entre o utilizador e a aplicação durante o início da sessão. Quando o utilizador liga a aplicação, esta aplicação disponibiliza o ecrã do PatternLock Scrambler como meio de autenticação. Assim que autenticado, o PatternLock Scrambler passa à aplicação o resultado (utilizador legítimo) e a aplicação disponibiliza a sua informação diretamente ao utilizador. Enquanto a sessão for mantida, a aplicação e o

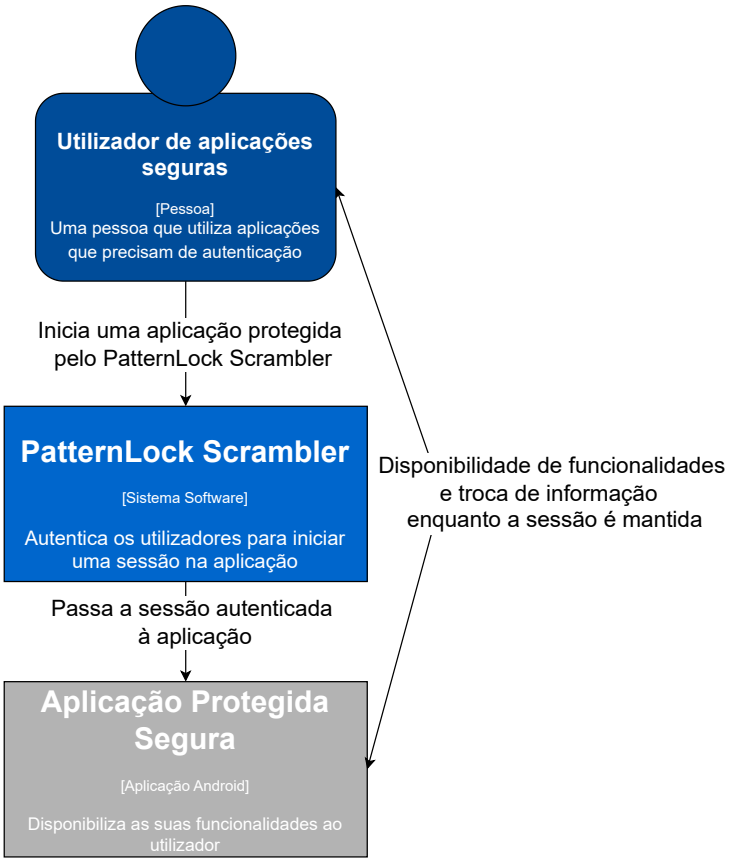


Figura 3: Nível 1 do modelo C4 da arquitetura da aplicação

utilizador comunicam diretamente como é esperado, pois o PatternLock Scrambler só é reativado assim que a sessão é terminada.

Na Figura 4, é apresentado o segundo nível da arquitetura do sistema e como os sistemas e a base de dados interagem.

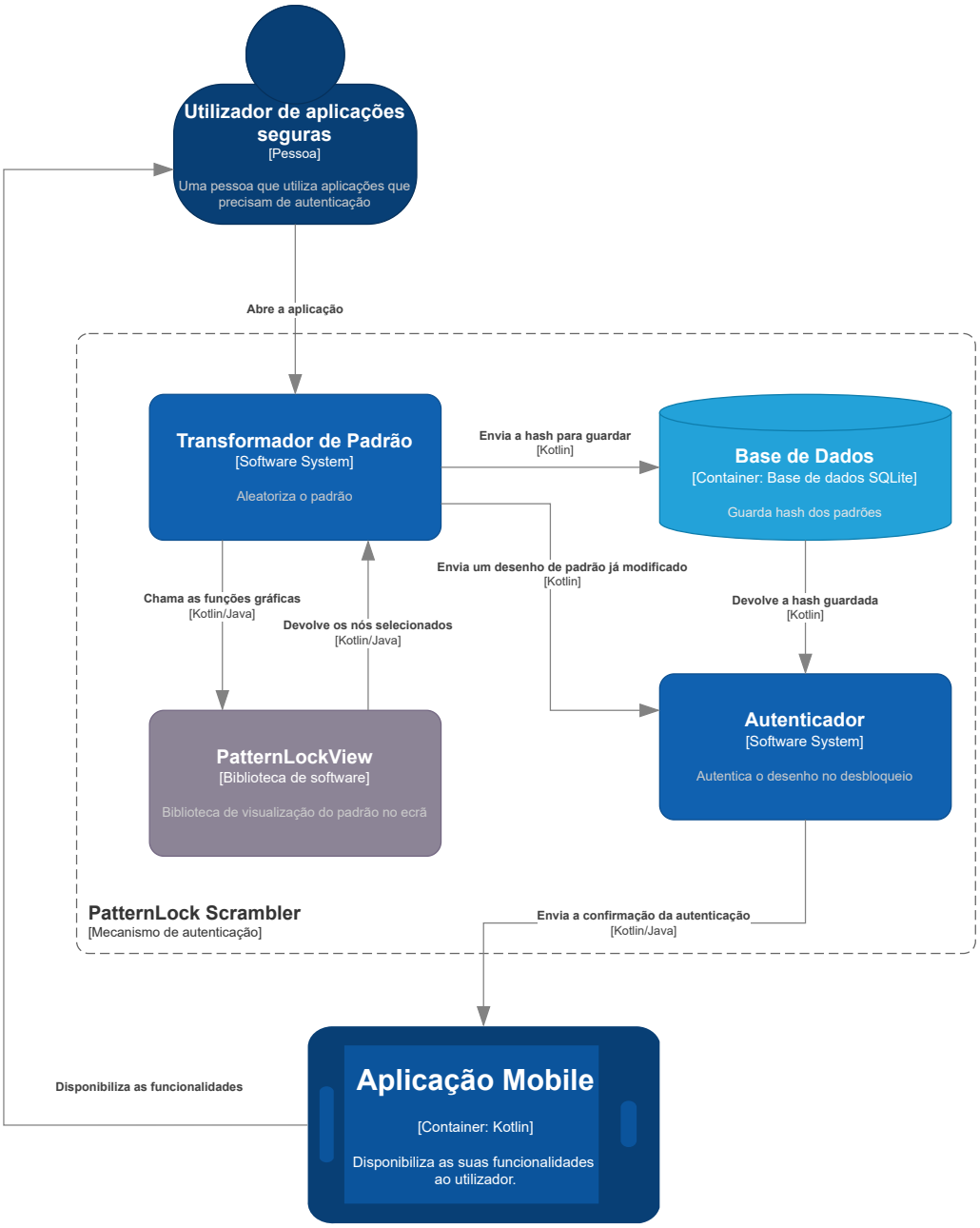


Figura 4: Nível 2 do modelo C4 da arquitetura da aplicação

Dentro do PatternLock Scrambler, só há 3 sistemas e a base de dados, como visto na Figura 4. O contexto do próprio programa e que calcula as operações matemáticas para aleatorizar o padrão, a base de dados onde estão guardadas as *hash* do padrão definido originalmente e que são usadas para validar o resultado do desenho, e o sistema que valida se a tentativa é válida e verdadeira.

Quando o utilizador liga a aplicação é imediatamente apresentado o ecrã de autenticação do padrão, este ecrã é convocado pelo transformador invocando a

biblioteca *PatternLockView*, a biblioteca regista os nós selecionados e devolve ao transformador. Se for a página de definições onde se define o padrão original, este é enviado para a base de dados sem alteração, caso contrário, o *scrambler* transpõe o padrão e envia para o autenticador que, através de uma função de hashing, calcula o *message digest* do padrão utilizado e compara à *hash* guardada na base de dados. No caso de serem idênticas, é enviada a confirmação para a aplicação. O utilizador e a aplicação interagem no início aquando o utilizador a inicia e depois quando a autenticação é validada e começa a utilização normal.

A especificação da interação dos sistemas está na Figura 5.

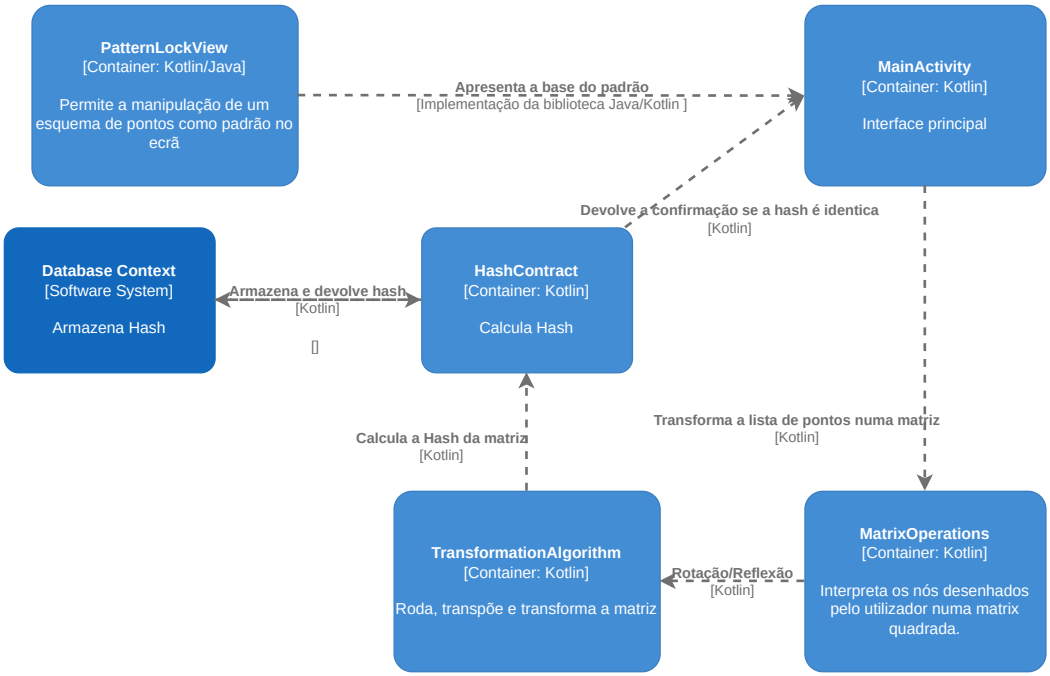


Figura 5: Nível 3 do modelo C4 da arquitetura da aplicação

A natureza do sistema requer que haja interação pelo utilizador, logo, ao iniciar a aplicação, o utilizador deparar-se-á com uma página onde pode desenhar o padrão de autenticação para se validar. Esta é a primeira atividade.

Seguindo, a segunda atividade é aquela em que o utilizador define o padrão original para se usar como comparação. Esta atividade foi desenhada para servir apenas como uma página de definições onde o utilizador pode alterar certas especificações do sistema ao seu gosto.

As principais alterações são as operações que afetam os esquemas desenhados pelo utilizador e contam com: rotação de pontos, reflexão dos pontos, e calculo de um *message digest* ou *hash* [1] do esquema realizado.

O sistema está dividido em 6 "áreas": A *MainActivity/interface* de utilizador é onde o utilizador interage com o sistema ao desenhar os padrões e personalizar as definições. Depois, na *PatternLockView* há a parte de reconhecimento do padrão, esta é feita por meio de uma biblioteca previamente desenvolvida por outros autores. No *TransformationAlgorithm* está presente a transformação com base no tempo (hora e minuto), esta parte essencial é a base deste projeto, nesta área estão todos os algoritmos e funções que transformam um padrão inicial num novo esquema. A realização da transformação dos esquemas faz parte do *MatrixOperations* que transforma o padrão criado num identificador único que pode ser posteriormente guardado e comparado mais facilmente. Na *Database Context*, ou seja, a área da base de dados, devido ao sistema ser desenvolvido em Android, e por não haver considerações a ligações por web, está situada a base de dados SQLite local onde é guarda a *hash* do padrão definido como base pelo utilizador. Finalmente, na fase *HashContract* em que se realiza a comparação de *hashes* para validar o utilizador, o sistema calcula a *hash* final do padrão transformado, obtém o *hash* guardado e compara ao *hash* do padrão agora calculado, e assim, caso sejam idênticos, autoriza o utilizador.

3.2 CONCEITO GERAL

O software concebido tem o propósito de autenticar utilizadores a partir de desenhos de padrões e, ao mesmo tempo, evitar o *shoulder-surfing* que pode invalidar as defesas estabelecidas por este método. O mecanismo aqui concebido aspira reduzir as capacidades de *shoulder-surfing* através de transformações do padrão desenhado de modo que, para um atacante, pareça ser sempre um padrão distinto, mas que, no entanto, é, na verdade, apenas uma variação do padrão original que apenas o verdadeiro utilizador tem conhecimento.

Para gerar modificações do padrão, o programa utiliza *timestamps*, ou horas e minutos, como uma base donde pode retirar um modificador determinístico do padrão que o utilizador também pode ver. O conceito do programa conta com

1 Um *hash* é o resultado de uma função que, a partir de qualquer *input*, consegue gerar um único *output* determinístico distinto de tamanho constante de tal forma que, a partir do *output* gerado, seja impraticável extrair o *input* original. Em outras palavras, *hash function* é uma função de um único sentido [61], [62].

rotações e reflexões do padrão desenhado para o transformar, mas permite que o utilizador, ao aceder as definições, tenha a possibilidade de alterar a forma em que o padrão é alterado, dentro de certos limites.

A verificação do utilizador foi concebida de maneira a funcionar pela comparação de *hashes*, uma destas guardada na base de dados, e a outra gerada pelo utilizador no momento de entrada na aplicação. É desta maneira que o software visa proteger a credencial do utilizador.

3.3 DEFINIÇÃO

Para possibilitar a transformação dos pontos foi necessário construir um esquema de código que traduz um ponto selecionado numa matriz de pontos para que esta possa ser alterada.

Primeiro passo é leitura do padrão desenhado. Depois da captura do padrão, cada nó selecionado é adicionado por ordem a uma matriz. Com esta matriz construida os algoritmos de transformação já podem rodar e espelhar o esquema desenhado, calcular a *hash* e validar. A matriz é transformada conforme as definições escolhidas pelo utilizador e pode ser espelhada e/ou rodada, conforme a *timestamp* do momento de autenticação.

3.3.1 *Leitura do padrão*

A leitura é feita por meio de números identificadores de cada nó selecionável, estes números vão de 0 a n^2-1 sendo "n" a largura do padrão em número de nós, visto que o padrão terá sempre lados iguais. A Tabela 1 apresenta os 'IDs' dos nós num padrão 3x3.

Ao selecionar um nó, o número associado a esse nó é adicionado a uma lista temporária, que, assim que o utilizador acabar o desenho, utiliza técnicas matemáticas para converter numa matriz quadrada de lado "n".

Com a utilização da biblioteca anterior é possível atribuir a uma variável a lista inteira, então, assim é enviada a variável para uma função que passa por todos os elementos da lista e adiciona a um *Array* bidimensional a posição na lista de cada nó para que, visualmente, seja possível reconhecer uma matriz de lado "n" com a ordem de seleção dos nós.

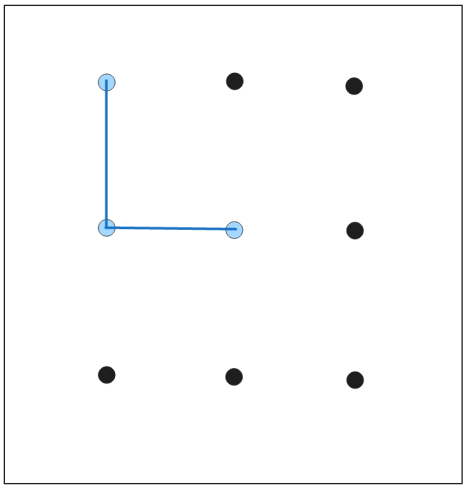


Figura 6: Desenho de um padrão que passa pelo nó 0, 3 e 4.

0	1	2
3	4	5
6	7	8

Tabela 1: Identificadores dos nós num padrão 3×3

Utilizando o exemplo da Figura 6, o qual é um desenho de padrão que começa pelo nó ID 0, passa pelo 3 e acaba no 4, a matriz resultante do desenho será Tabela 2. Para efeitos dos cálculos, na matriz resultante, os 0 são nós não escolhidos.

Agora, a partir da matriz resultante é possível notar que, no caso da Figura 6, o primeiro nó selecionado foi o do canto superior esquerdo, a seguir foi o meio esquerdo e o final foi o centro, como se pode observar na Tabela 2.

1	0	0
2	3	0
0	0	0

Tabela 2: Resultado da transformação da lista de IDs dos nós selecionados numa matriz que indica a ordem de seleção dos nós

3.3.2 *Transposição da matriz*

0	2	1
0	3	0
0	0	0

Tabela 3: Resultado de uma rotação de 90.º à matriz inicial.

A partir da Tabela 2 agora é possível visualizar mentalmente as transformações ao padrão. Estas operações são a base de todas as transformações realizadas. Como, por exemplo, se se realizar uma rotação de 90.º (no sentido do relógio) ao padrão, a matriz resultante será a Tabela 3.

O que a Tabela 3 demonstra é que, após a rotação, o primeiro nó selecionado passou a ser o nó no canto superior direito (ID 2), depois o centro superior (ID 1) e no final o centro (ID 4).

Em específico, roda uma dada matriz 2D (representada como um *array* de *arrays*) 90 graus no sentido dos ponteiros do relógio.

Primeiro obtém o número de linhas e colunas da matriz inicial. Depois cria uma matriz resultado vazia com as dimensões da matriz de entrada transpostas (ou seja, as linhas tornam-se colunas e as colunas tornam-se linhas) e itera através dos elementos da matriz inicial.

Para cada elemento, calcula a sua nova posição na matriz rodada, trocando os seus índices de linha e coluna e invertendo o índice de linha e coloca o elemento na posição correspondente na matriz resultante.

0	0	0
0	3	0
0	2	1

Tabela 4: Resultado de uma reflexão horizontal da matriz resultante da rotação.

Para espelhar a matriz, o algoritmo troca as linhas de posição para espelhar horizontalmente, no caso de espelhar verticalmente então troca as colunas de posição.

Em primeiro lugar obtém o número de linhas e colunas da matriz inicial. A seguir, cria uma matriz de resultados vazia com as mesmas dimensões que a matriz inicial e itera pelos elementos da matriz de entrada.

Para cada elemento ciclável calcula a sua nova posição na matriz espelhada, invertendo o seu índice de coluna e coloca o elemento na posição correspondente na matriz resultante. No final da operação, a matriz resultante será a Tabela 4, no caso de espelhar horizontalmente.

O sistema guarda a *timestamp* no momento de toque no primeiro nó de cada desenho de padrão para os efeitos de transposição. Como o utilizador é que define o tipo de operações a serem utilizadas, então é assumido pelo sistema que esse mesmo manterá na sua memória quais operações são realizadas em cada caso.

O sistema, por razões de segurança, não apresenta nem indica ao utilizador quais as operações a realizar no momento de autenticação, pois isso iria contra o propósito de evitar ataques *shoulder-surfing*.

Figura 7: *Mockup* da página de definições onde o utilizar define qual operação para cada hora/minuto

Numa página própria de definições, o utilizador decide qual operação será executada para cada intervalo de hora/minuto. Para cada opção escolhida, uma opção desaparecerá da lista próxima de operações para não repetir as mesmas. A Figura 7

é uma representação da página de definições, onde o utilizador pode escolher cada operação para a hora e minuto.

Conforme demonstrado na Figura 7, o utilizador escolheu a hora para a operação "Rotação de 90º no sentido do relógio". Sendo assim, quando o utilizador desbloquear o dispositivo numa hora par, a operação será a mesma, enquanto, como na hora ímpar foi escolhida a rotação contra o relógio, nessas horas seria uma operação contrária.

Se considerarmos que as opções escolhidas por um potencial utilizador são as mesmas apresentadas na Figura 7, e que as Tabelas 3 e 4 são representações de uma autenticação real, então agora é possível extrair que o utilizador autenticou-se num momento com a hora e os minutos pares.

3.4 BASE TECNOLÓGICA

Nesta secção será apresentada uma breve panorâmica das principais tecnologias e ferramentas utilizadas nesta investigação. Este trabalho exige um conhecimento profundo destas tecnologias para que se possa apreciar a metodologia, os resultados e os contributos da investigação.

As subsecções seguintes abordam as principais tecnologias utilizadas nesta investigação, incluindo o seu desenvolvimento histórico, princípios gerais e relevância para o domínio do desenvolvimento de mecanismos de segurança. Estas informações de *background* seguintes permitirão uma melhor compreensão das aplicações e adaptações específicas destas tecnologias na investigação atual.

3.4.1 *Android*

A escolha do sistema Android para sistema operativo deste projeto deve-se ao fato de que é o sistema *mobile* mais utilizado no mundo [63], como pode ser visto na Figura 8, é *open-source* [64] e disponibiliza diversas ferramentas aos programadores, como um emulador[2] de dispositivo Android.

2 A plataforma Android Studio, o ambiente de desenvolvimento oficial para Android, permite que uma pessoa crie um dispositivo virtual para os efeitos de teste [65].

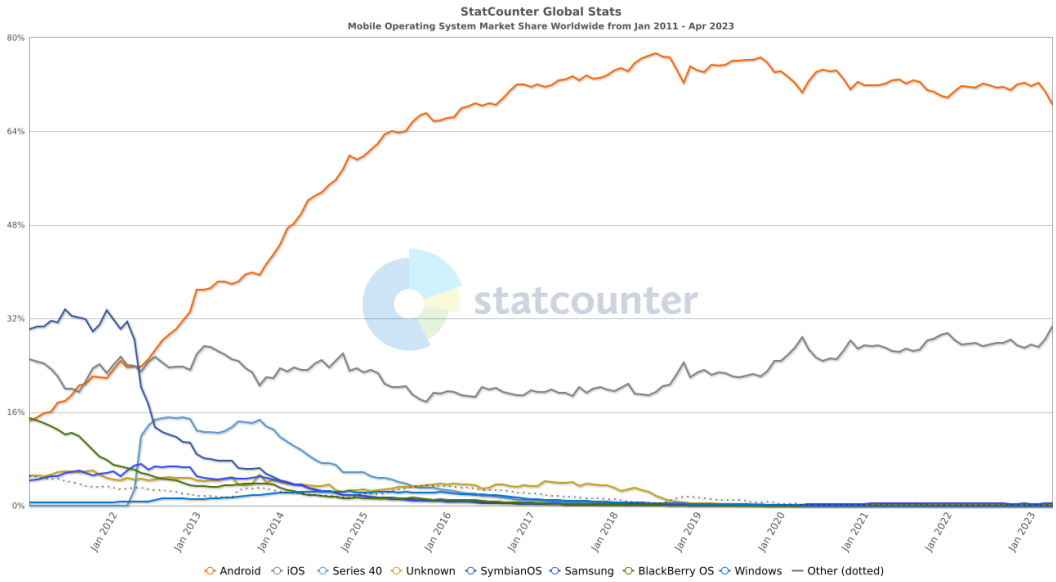


Figura 8: Quota de mercado mundial dos sistemas operativos móveis entre janeiro de 2011 - abril de 2023 [1]

3.4.2 *Kotlin*

As opções disponíveis para a linguagem deste projeto eram entre Kotlin e Java. A linguagem Kotlin foi a escolhida devido ao software desenvolvido ser principalmente virado ao sistema Android, porque é mais legível que Java, tem *NullSafety*, é interoperável com Java, funciona adequadamente com as bibliotecas escritas em Java, é desenvolvido mais ativamente, entre outras funcionalidades mais modernas [19], [66].

3.4.3 *SQLite*

Para guardar informação sobre a aplicação, este projeto utilizou uma base de dados SQLite. SQLite é uma biblioteca que implementa um motor de base de dados regularmente utilizado em aplicações móveis por ser leve, rápido, autónomo e com todas as funcionalidades necessárias para armazenamento local. Esta tecnologia foi escolhida por ser rápida e simples de utilizar, por não adicionar um *overhead* mais pesado à aplicação e por ser facilmente integrável numa aplicação *mobile* [67]. O propósito de utilização do motor SQLite no PatternLock Scrambler é a organização e armazenamento de várias *digests* na mesma tabela de uma forma indexada.

3.4.4 *Tecnologias adicionais*

Para expedir o desenvolvimento do software e poder focar no ponto-chave deste projeto foi utilizada uma biblioteca pública *open-source* de criação de uma interface visual onde desenhar os padrões de autenticação.[3]

Esta biblioteca é baseada na biblioteca da Android Open Source Project (AOSP), mas com umas alterações que aceleram o desenvolvimento do projeto. As principais vantagens desta biblioteca são: é possível traduzir os nós selecionados numa *String* de valores, é altamente personalizável em termos de tamanho de padrão, cores, efeitos, etc. Adicionalmente, também é possível modificar o eXtensible Markup Language (XML) programaticamente.

O domínio sob a *interface* de padrão oferecida por esta biblioteca é o que torna esta uma base excelente para começar a desenvolver.

Antes de escrever os algoritmos de transformação de matizes foram investigadas alternativas nativas ao Kotlin, tais como opengl/Matrix[4], graphics/Matrix[5] e a Multik[6]. As bibliotecas OpenGL/Matrix e graphics/Matrix, já incorporadas à linguagem Kotlin, foram implementadas no início do projeto, visto que já implementavam funções de rotação. Contudo, devido à sua natureza estática nas dimensões, 4 e 3, respetivamente, que impede a utilização de padrões com larguras/alturas superiores a 4 ou inferiores a 3, foram rejeitadas. A biblioteca Multik foi analisada por fim, mas não possui métodos de rotação nem reflexão, o que a levou a ser rejeitada.

3 https://github.com/aritraroy/PatternLockView
4 https://developer.android.com/reference/kotlin/android/opengl/Matrix
5 https://developer.android.com/reference/kotlin/android/graphics/Matrix
6 https://blog.jetbrains.com/kotlin/2021/02/multik-multidimensional-arrays-in-kotlin/

4

DESENVOLVIMENTO

Neste capítulo é apresentado o desenvolvimento prático da aplicação Android usada utilizada como produto viável mínimo. Este capítulo também explica tanto a lógica atrás dos algoritmos que alteram o padrão introduzido pelo utilizador, como o fluxo de um exemplo de uso específico.

4.1 PRODUTO VIÁVEL MÍNIMO

Neste projeto, desenvolveu-se uma aplicação Android em Kotlin chamada *Pattern-Lock Scrambler* que tem as seguintes funcionalidades: permite que o utilizador disfarce o seu padrão de desbloqueio por meio de operações de transformação do padrão de uma maneira previsível para o utilizador, mas sempre diferente para um atacante; permite que o utilizador personalize as operações que deseja para cada hora/minuto.

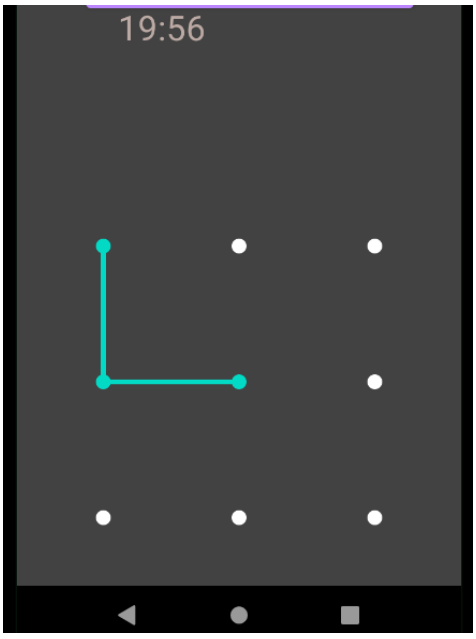


Figura 9: Ecrã de autenticação inicial

Ao iniciar a aplicação, o utilizador encontrará um ecrã com o método de autenticação desenvolvido. O utilizador desenha o padrão transformado consoante as respetivas transposições. Se o utilizador ainda não tiver definido o seu padrão de autenticação base, pode acessar a página de definições para alterar para o que desejar.

1	2	0
0	3	0
0	0	0

Tabela 5: Representação da ordem de seleção de nós aquando autenticação numa hora ímpar e minuto par, baseando-se no padrão anterior

Usando a Tabela 2 como exemplo de padrão, como mostra a Figura 9, um utilizador que se autenticasse às 19:56 (hora ímpar e minuto par) teria que desenhar um padrão representado por Tabela 5, ou seja, uma rotação de 90º contra o relógio, seguida de uma reflexão horizontal.

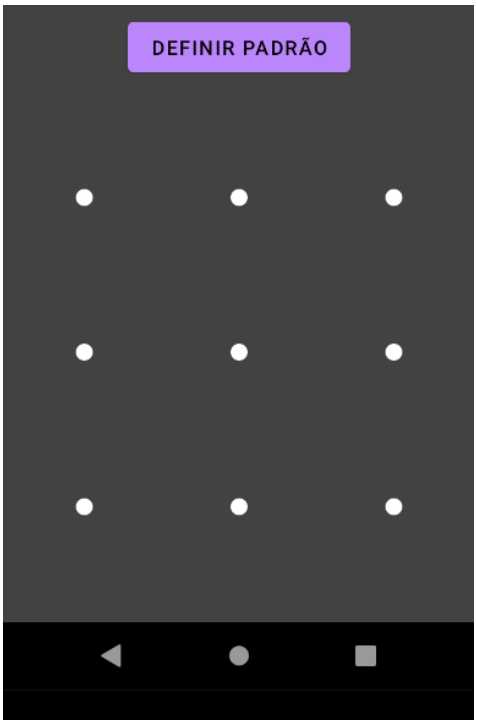


Figura 10: Página de definições

A página de definições, enunciada na Figura 10, é uma atividade que permite que o utente proponha o padrão a ser utilizado.

4.2 IMPLEMENTAÇÃO

A aplicação está dividida em 3 partes: interface gráfica e código relevante; funções de criação de matrizes e transformação das mesmas, e código relevante à base de dados. A interface gráfica apresenta o padrão inicial, o código relevante a esta atividade recebe *input* do utilizador durante o desenho e recebe input do relógio do sistema operativo.

Durante as primeiras etapas do projeto, era criado um código completamente original para gerar um ecrã com um padrão onde se pode interagir e selecionar nós. Contudo, após uma análise das tecnologias existentes, encontrou-se a biblioteca da AOSP, onde se baseia a *PatternViewLock*, escolha definitiva para este projeto.

O código relevante ao padrão tem o propósito de apresentar no ecrã o padrão de uma forma dinâmica, de guardar em memória todos os nós selecionados e de os devolver a outra função no fim do desenho do padrão. A função que recebe o desenho é uma função que substitui a função nativa da biblioteca que traduz a lista de nós selecionados numa *String* de valores. A função da biblioteca é substituída porque originalmente agrega todos os 'IDs' dos nós seguidamente sem quaisquer caracteres a diferenciar os mesmos e por isso, em casos que o ecrã de padrão seja uma figura maior do que $3 \cdot 3$, é necessário adicionar caracteres separadores de valores. Por exemplo, antes de trocar a função, um desenho de padrão exemplar devolveria em formato *String* apenas o valor "310". Em esquemas de $3 \cdot 3$, não há problemas em distinguir entre 'IDs' dos nós, pois o maior 'ID' é 8. No entanto, em esquemas $4 \cdot 4$ ou maiores, este mesmo padrão geraria uma diferença entre ter o padrão sequencial "3; 1; 0", e o padrão "3; 10".

A função original foi, portanto, modificada para adicionar o caratere (,) entre cada 'ID', de forma que uma "String" de valores resultantes de desenho de padrão agora será parecida com "3,1,0" em vez de "310". A nova técnica permite obter valores exatos da *String* sem ambiguidade e seguidamente traduzidos numa matriz para uso subsequente.

Finalizando a interface do utilizador *MainActivity*, é a partir daqui que são invocadas as funções que transformam a lista de valores numa matriz e transpõem essa mesma para equivaler a *hash* do padrão definido antecipadamente.

Relativamente às funções da matriz, criou-se uma função que traduz a lista[1] inicial de valores em formato de matriz. Esta função percorre todos os índices da lista, verifica qual o valor em cada um e, consoante as dimensões da área de desenho

1 List<Int>, em que 'Int' é o 'ID' do nó pressionado

e do valor selecionado, insere numa matriz de 0s qual a ordem de seleção do nó na posição representante da interface gráfica.

```kotlin
fun listToMatrix(list: List<Int>,
                 rows: Int,
                 cols: Int
): Array<Array<Int>> {
    val matrix = Array(rows) { Array(cols) { 0 } }

    for (i in list.indices) {
        val position = list[i]
        val row = position / rows
        val col = position % cols
        matrix[row][col] = i + 1
    }
}
```

Listagem 1: Código-fonte Kotlin que transforma uma lista numa matriz

A função `listToMatrix` apresentada na Listagem 1 recebe `list`, o número de linhas (`rows`) e o número de colunas (`cols`) como parâmetros. Inicializa uma matriz 2D chamada `matrix` com zeros usando `Array(rows) Array(cols) 0`. A função itera sobre os índices da `list` usando `for (i in list.indices)`. Dentro de cada iteração, a função obtém a `position` subtraindo 1 do valor em `list[i]`. A ordem de seleção do utilizador é alterada para um índice de zero.

O índice da linha é calculado usando `val row = position / cols`. Estabelece a linha onde o elemento deve ser posicionado, consoante a posição dividida pelo número de colunas.

O índice da coluna é calculado usando `val col = position % cols`. Determina a coluna na linha com base no resto (módulo) da posição dividida pelo número de colunas.

Por fim, a função atribui a ordem de seleção (`i + 1`) à posição correspondente na `matrix` usando `matrix[row][col] = i + 1`.

A função `listToMatrix` devolve a `matrix` resultante assim que todos os elementos forem atribuídos.

Na função `main`, invocamos `listToMatrix` com `patternList`, `cols` e `cols` (já que a matriz é quadrada). A matriz resultante é armazenada na variável `matrix`.

```kotlin
fun rotate90Degrees(matrix: Array<Array<Int>>): Array<Array<Int>> {
    val rows = matrix.size
    val cols = matrix[0].size
    val result = Array(cols) { Array(rows) { 0 } }
    for (i in 0 until rows) {
        for (j in 0 until cols) {
            result[j][rows - i - 1] = matrix[i][j]
        }
    }
    return result
}
```

Listagem 2: Código-fonte Kotlin que roda a matriz em 90º

O código na Listagem 2 define uma função chamada `rotate90Degrees` que recebe uma matriz de entrada representada como um *array* bidimensional de inteiros (`Array<Array<Int>>`) e devolve uma nova matriz que é a rotação de 90 graus no sentido horário da matriz original.

A função `rotate90Degrees` aceita a matriz de entrada como parâmetro: `matrix: Array<Array<Int>>`. A matriz é esperada como um *array* bidimensional onde cada elemento representa um número inteiro.

As próximas duas linhas de código determinam o número de linhas (`rows`) e colunas (`cols`) na matriz de entrada usando as propriedades `size`. Pressupõe-se que a matriz não está vazia e que todas as linhas têm o mesmo número de colunas.

O código inicializa uma nova matriz bidimensional chamada `result` com as dimensões invertidas da matriz de entrada. Utiliza-se o construtor `Array`, especificando o número de colunas (`cols`) e linhas (`rows`). Cada elemento da matriz `result` é inicializado com o valor "0" inicialmente.

O próximo *loop for* itera sobre as linhas da matriz de entrada. Utiliza-se a variável de índice `i` para iterar o número da linha atual, começando em 0 e indo até (mas não incluindo) o valor de `rows`.

No *loop* externo, há outro *loop for* que itera sobre as colunas da matriz de entrada. Utiliza-se a variável de índice `j` para iterar o número da coluna atual, começando em 0 e indo até (mas não incluindo) o valor de `cols`.

No interior do segundo *loop for*, o código atribui o valor do elemento `i`-ésima linha e `j`-ésima coluna da matriz de entrada ao elemento correspondente da matriz `result`, mas com as coordenadas trocadas e a linha espelhada. Isso é feito utilizando a expressão `result[j][rows - i - 1] = matrix[i][j]`.

Após a conclusão dos *loops*, a matriz `result` contém a rotação de 90 graus no sentido horário da matriz de entrada.

Por fim, a matriz `result` é devolvida como resultado da função.

```
1  fun rotate270Degrees(matrix: Array<Array<Int>>): Array<Array<Int>> {
2
3      val rows = matrix.size
4      val cols = matrix[0].size
5
6      val result = Array(cols) { Array(rows) { 0 } }
7      for (i in 0 until rows) {
8          for (j in 0 until cols) {
9              result[cols - j - 1][i] = matrix[i][j]
10         }
11     }
12     return result
13 }
```

Listagem 3: Código-fonte Kotlin que roda a matriz em -90º

O código na Listagem 3 define uma função chamada `rotate270Degrees` que recebe uma matriz de entrada representada como um *array* bidimensional de inteiros (`Array<Array<Int>>`) e devolve uma nova matriz que é a rotação de 90 graus no sentido horário da matriz original.

A função `rotate270Degrees` aceita a matriz de entrada como parâmetro: `matrix: Array<Array<Int>>`. A matriz é esperada como um *array* bidimensional onde cada elemento representa um número inteiro.

As próximas duas linhas de código determinam o número de linhas (`rows`) e colunas (`cols`) na matriz de entrada usando as propriedades `size`. Pressupõe-se que a matriz não está vazia e que todas as linhas têm o mesmo número de colunas.

O código inicializa uma nova matriz bidimensional chamada `result` com as dimensões invertidas da matriz de entrada. Utiliza-se o construtor `Array`, especificando o número de colunas (`cols`) e linhas (`rows`). Cada elemento da matriz `result` é inicializado com o valor "0" inicialmente.

O próximo *loop for* itera sobre as linhas da matriz de entrada. Utiliza-se a variável de índice `i` para iterar o número da linha atual, começa em 0 e vai até (mas não incluindo) o valor de `rows`.

No *loop* externo, há outro *loop for* que itera sobre as colunas da matriz de entrada. Utiliza-se a variável de índice `j` para iterar o número da coluna atual, começa em 0 e vai até (mas não incluindo) o valor de `cols`.

No interior do segundo *loop for*, o código atribui o valor do elemento `i`-ésima linha e `j`-ésima coluna da matriz de entrada ao elemento correspondente da matriz `result`, mas com as coordenadas trocadas e a linha espelhada. Isso é feito utilizando a expressão `result[cols - j -1][rows] = matrix[i][j]`.

Após a conclusão dos *loops*, a matriz `result` contém a rotação de 270 graus no sentido horário da matriz de entrada.

Por fim, a matriz `result` é devolvida como resultado da função.

```
1 fun mirrorHorizontally(matrix: Array<Array<Int>>): Array<Array<Int>> {
2     val rows = matrix.size
3     val cols = matrix[0].size
4     val result = Array(rows) { Array(cols) { 0 } }
5     for (i in 0 until rows) {
6         result[i] = matrix[rows - i - 1]
7     }
8     return result
9 }
```

Listagem 4: Código-fonte Kotlin que reflete a matriz horizontalmente

O código na Listagem 4 apresenta uma função denominada `mirrorHorizontally`, que recebe uma matriz de entrada representada por um *array* bidimensional de inteiros `Array<Array<Int>>` e devolve uma nova matriz, a qual é um espelho horizontal da matriz original.

A função `mirrorHorizontally` recebe a matriz de entrada como parâmetro: `matrix: Array<Array<Int>>`. A matriz é esperada como um *array* bidimensional onde cada elemento representa um número inteiro.

As próximas duas linhas de código determinam o número de linhas (`rows`) e colunas (`cols`) na matriz de entrada usando a propriedade `size`. É assumido que a matriz não está vazia e todas as linhas têm o mesmo número de colunas.

O código inicializa uma nova matriz bidimensional chamada `result` com as mesmas dimensões da matriz de entrada. Ele usa o construtor `Array`, especificando o número de linhas (`rows`) e colunas (`cols`). Cada elemento da matriz `result` é inicializado como "0" inicialmente.

O próximo *loop for* itera sobre as linhas da matriz de entrada. Este usa a variável de índice `i` para iterar o número da linha atual, começando em 0 e indo até (mas não incluindo) o valor de `rows`.

Dentro do *loop*, o código atribui a linha `i` da matriz `result` acessando a linha correspondente da matriz de entrada, mas em ordem reversa. Isso é feito usando a

expressão `matrix[rows - i - 1]`. A expressão `rows - i - 1` calcula o índice da linha a ser espelhada.

Após o término do *loop*, a matriz `result` é a versão espelhada horizontalmente da matriz de entrada. Por fim, a matriz `result` é devolvida como saída da função.

```
1  fun mirrorVertically(matrix: Array<Array<Int>>): Array<Array<Int>> {
2      val rows = matrix.size
3      val cols = matrix[0].size
4      val result = Array(rows) { Array(cols) { 0 } }
5      for (i in 0 until rows) {
6          for (j in 0 until cols) {
7              result[i][j] = matrix[i][cols - j - 1]
8          }
9      }
10     return result
11 }
```

Listagem 5: Código-fonte Kotlin que reflete a matriz verticalmente

O código que apresentado na Listagem 5 define uma função chamada `mirrorVertically` que recebe uma matriz de entrada representada como um *array* bidimensional de inteiros (`Array<Array<Int>>`) e devolve uma nova matriz que é um espelho vertical da matriz original.

Aqui está uma explicação passo-a-passo do código:

A função `mirrorVertically` aceita a matriz de entrada como parâmetro: `matrix: Array<Array<Int>>`. A matriz é esperada como um *array* bidimensional onde cada elemento representa um número inteiro.

As próximas duas linhas de código determinam o número de linhas (`rows`) e colunas (`cols`) na matriz de entrada usando as propriedades `size`. Pressupõe-se que a matriz não está vazia e todas as linhas têm o mesmo número de colunas.

O código inicializa uma nova matriz bidimensional chamada `result` com as mesmas dimensões da matriz de entrada. Utiliza-se o construtor `Array`, especificando o número de linhas (`rows`) e colunas (`cols`). Cada elemento da matriz `result` é inicializado com o valor "0" inicialmente.

O próximo *loop for* itera sobre as linhas da matriz de entrada. Utiliza-se a variável de índice `i` para iterar o número da linha atual, começando em 0 e indo até (mas não incluindo) o valor de `rows`.

No *loop*, há outro *loop for* que itera sobre as colunas da matriz de entrada. Utiliza-se a variável de índice `j` para iterar o número da coluna atual, começando em 0 e indo até (mas não incluindo) o valor de `cols`.

No interior do segundo *loop for*, o código atribui o valor do elemento `i`-ésima linha e `j`-ésima coluna da matriz `result` ao elemento correspondente da matriz de entrada, mas na ordem inversa relativamente às colunas. A operação é feita com a expressão `matrix[i][cols - j - 1]`. A expressão `cols - j - 1` calcula o índice da coluna a ser espelhada.

Após a conclusão dos *loops*, a matriz `result` contém a versão espelhada verticalmente da matriz de entrada. Por fim, a matriz `result` é devolvida como resultado da função.

Finalmente, as funções/atividades da base de dados correspondem a código presente na maioria de aplicações Kotlin que recorrem a uma base de dados SQLite como persistência de dados. Estas funções existem para criar uma base de dados onde guardar a *hash* e para poder obter esse valor a qualquer momento sem ter de recalcular a *hash* original.

4.2.1 *Exemplo de uso*

Um exemplo de utilização é agora apresentado, em que a Figura 11 apresenta o fluxo de instruções no momento de autenticação. Utilizando um padrão definido com trajeto 5; 4; 6; ou seja, um desenho igual à Figura 12b.

Como exemplo, a hora de autenticação é de 13:08. Dessa forma, tendo em vista que a hora e o minuto são pares, as instruções são rodar o padrão em 90º e espelhar o mesmo verticalmente. O diagrama Figura 11 apresenta as operações que a aplicação faz com as definições iguais e às 13:08.

Para autenticar às 13:08 então o utilizador deverá desenhar o padrão Figura 13, que irá, através da função na Listagem 2, ser rodado para ser Figura 14a. Por fim, o sistema espelha o mesmo com ajuda da Listagem 5 para obter o padrão Figura 14b, depois verificará para ver se é igual a Figura 12a. Como os padrões são idênticos, o utilizador é autenticado.

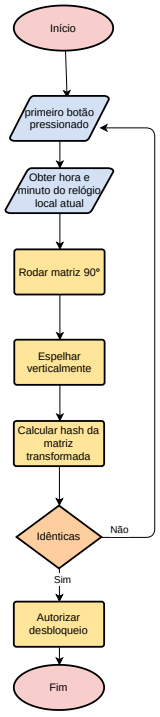
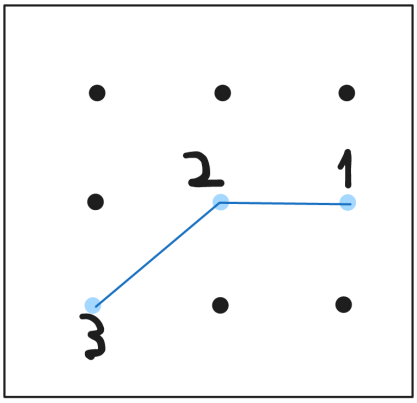


Figura 11: Fluxo de Instruções

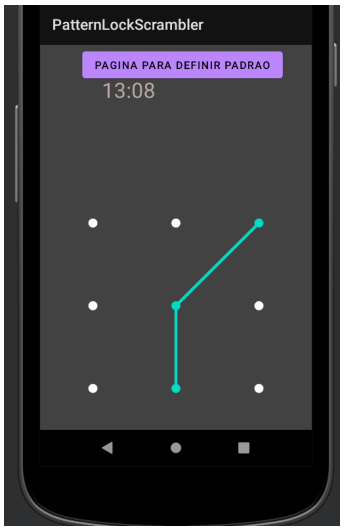
As alterações do padrão são feitas **internamente** e como tal nunca são reveladas ao utilizador. Neste exemplo são apresentados todos os passos de transformação para ilustrar os cálculos internos da aplicação.

(a) Definição de padrão a utilizar como exemplo.

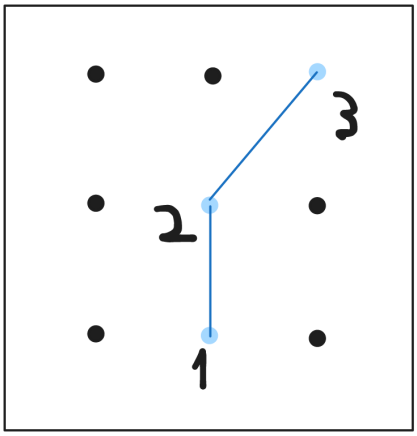


(b) Padrão original guardado na app.

Figura 12: Definição do padrão

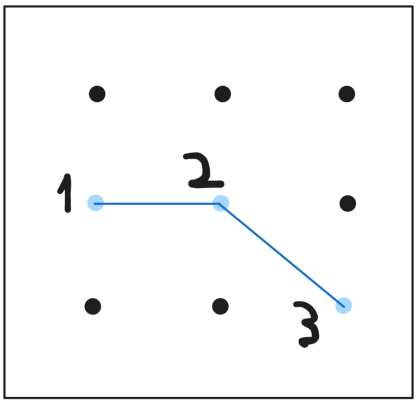


(a) Desenho inicial aquando autenticação.

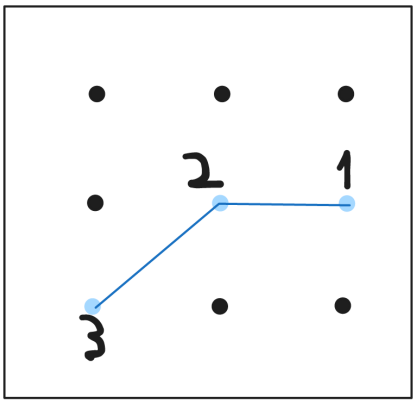


(b) Padrão desenhado no ecrã no momento de autenticação.

Figura 13: Momento de autenticação do ponto de vista do utilizador e o que é registado pelo *smartphone*



(a) Padrão rodado 90º.



(b) Padrão espelhado verticalmente.

Figura 14: Transformações aplicadas ao padrão pelo *smartphone*

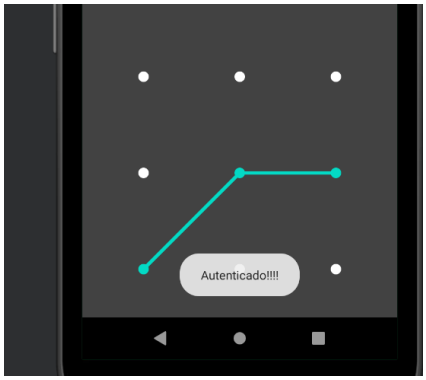


Figura 15: Resultado do desenho autentica o utilizador

4.3 SÍNTESE

Esta secção centrou-se na implementação de várias características e funcionalidades da aplicação Android utilizando Kotlin como linguagem de programação principal. Utilizando o poder e a flexibilidade de Kotlin, foi construída uma aplicação que funciona como prova de conceito para o esquema de autenticação introduzido.

O código-fonte desenvolvido nesta fase está disponível no GitHub[2].

2 https://github.com/RicardoCacao/PatternLockScrambler

5

TESTES

Neste capítulo, são apresentados os testes realizados e os seus resultados, as conclusões aprendidas a partir deles, bem como uma hipótese para explicar o motivo de tais resultados.

Tendo sido realizado este projeto no âmbito de mestrado, não houve tempo suficiente para realizar um teste real em produção, portanto, os testes são respondidos por utilizadores que apenas usaram a versão produto viável mínimo. O sistema desenvolvido neste projeto não foi embutido numa aplicação real, ou seja, não havia acesso a uma aplicação real onde implementar o mecanismo para utilização em casos reais.

Na ausência de testes reais, optou-se por elaborar testes de usabilidade do mecanismo. Estes testam não a capacidade e o funcionamento da aplicação, mas sim a experiência dos utilizadores e as opiniões dos mesmos. Assim, é possível avaliar a eficácia e a experiência do utilizador no contexto do sistema em desenvolvimento.

Esta estratégia permite identificar possíveis problemas com a usabilidade que os utilizadores podem enfrentar ao utilizar o sistema. Ao conduzir testes de usabilidade, os participantes foram solicitados a realizar uma autenticação real no sistema, permitindo ao observador capturar *feedback* direto sobre a facilidade de uso, a compreensão das funcionalidades e a identificação de áreas que podem necessitar de aprimoramentos.

Ao adotar testes de usabilidade na ausência de testes reais, a equipa de desenvolvimento teve a oportunidade de identificar antecipadamente possíveis obstáculos que os utilizadores podem enfrentar durante a utilização real do produto. Além disso, os testes de usabilidade permitiram recolher informação valiosa sobre a interface e a eficiência das interações, auxiliando na melhoria do design e na criação de uma experiência mais fluida e satisfatória para os utilizadores. Embora não substituam completamente os testes em ambiente real, os testes de usabilidade oferecem uma abordagem mínima para otimizar a usabilidade do produto antes da finalização do desenvolvimento.

5.1 INTRODUÇÃO

Foi construído um questionário[1] para receber as opiniões dos utilizadores acerca do sistema PatternLock Scrambler desenvolvido neste projeto.

O questionário apresenta 6 questões, cada uma destas aceita uma resposta de escala. As perguntas são as seguintes:

1. Quão importante é para si a autenticação em *smartphones*? - Escala de 1 (mínima) a 10 (extremamente importante)
2. Até que ponto se sente à vontade para aprender e adaptar-se a novas tecnologias ou métodos? - De 1(Nada à vontade) a 5 (Completamente à vontade)
3. Dificuldade de entender o Pattern Lock Scrambler - De 1 (Bastante Difícil) a 5 (Bastante Fácil)
4. Já encontrou dificuldades a autenticar - De 1 (Muitas) a 5 (Nenhuma)
5. Quantas vezes costuma tentar antes de ser autenticado - De 1 a 10 vezes
6. Quanto tempo está disposto a gastar no processo de autenticação de cada vez que desbloqueia o seu dispositivo? (Em segundos) - De 1 a 10 segundos

Este questionário visou dar a entender o sentimento das pessoas por novos métodos de autenticação e a aptitude das pessoas de testar um novo método.

O questionário foi preenchido pelos utilizadores após estes terem testado a usabilidade da aplicação.

Cada questão foi formulada com a intenção de disponibilizar maior quantidade de informação possível sobre o sistema testado. A primeira pergunta estabelece a disposição de uma pessoa no que consta as expetativas de segurança quando utiliza um *smartphone*, define a opinião de uma pessoa para com a autenticação e define uma base para melhor entendimento das perguntas seguintes.

Questão 2 esclarece quão confortável a pessoa se sente com aprender novas tecnologias para a sua segurança, se tem alguma aversão a mudar de mecanismo para um novo e estabelece uma relação entre a vontade de aprender e a dificuldade de entender o mecanismo na pergunta 3. Esta relação realça casos em que potenciais falhas do mecanismo são, na verdade, inerentes ao utilizador sentir-se pouco à vontade com novos métodos.

1 O questionário está presente no seguinte endereço: https://forms.gle/Sjdpt4qL8LEMz7t77

Questão 3 ajuda a perceber a adotabilidade do mecanismo no uso por utilizadores normais. Em combinação com a pergunta 2, esta questão destaca a dificuldade teórica da aprendizagem do método. Por exemplo: se uma pessoa completamente à vontade para aprender novas tecnologias responder que tem bastantes dificuldades a perceber o funcionamento do método (da parte virada ao utilizador) então essa resposta terá mais peso na classificação da usabilidade do sistema.

Questão 4 aspira avaliar a parte prática da utilização do método e se o utilizador encontra alguma dificuldade a utilizar o sistema.

Questão 5 mede objetivamente o número de tentativas que cada utilizador único realizou por cada autenticação válida. Esta questão foi concebida para poder avaliar a dificuldade prática da utilização. Valores altos nesta questão indicam que o método não é prático para uso diário, enquanto resultados baixos indicam o oposto.

A última questão percebe quanto tempo os utilizadores estão dispostos a gastar em cada autenticação válida. Respostas nesta questão ajudam a perceber se o método é aplicável a casos de uso onde é necessário aceder rapidamente ao *software* protegido pelo PatternLock Scrambler.

5.2 ANÁLISE DOS RESULTADOS

Apresentam-se a seguir as respostas recebidas pelo questionário:

Foram obtidas no total 20 respostas ao questionário. As respostas estão apresentadas nos gráficos das figuras: 16, 17, 18, 19, 20 e 21. As médias das respostas para cada pergunta recebidas são abreviadas para $\overline{P_n}$ em que "n" é o número da pergunta.

$\overline{P_1} = 5{,}35$; $\overline{P_2} = 3{,}15$; $\overline{P_3} = 3$; $\overline{P_4} = 2{,}8$; $\overline{P_5} = 6{,}2$; $\overline{P_6} = 6{,}1$;

A resposta mais escolhida para cada pergunta é:

$P_1 = 3$ e 5; $P_2 = 2$; $P_3 = 3$; $P_4 = 3$; $P_5 = 4$ e 8; $P_6 = 5$ e 10;

Das respostas recebidas é notável que apesar de a maioria das pessoas que responderam ao questionário se sentirem confortáveis com novas tecnologias e métodos (Figura 17), existe uma tendência de encontrar dificuldades a utilizar o mecanismo(Figura 20). Esta e outras constatações foram exploradas na Seção 5.3.

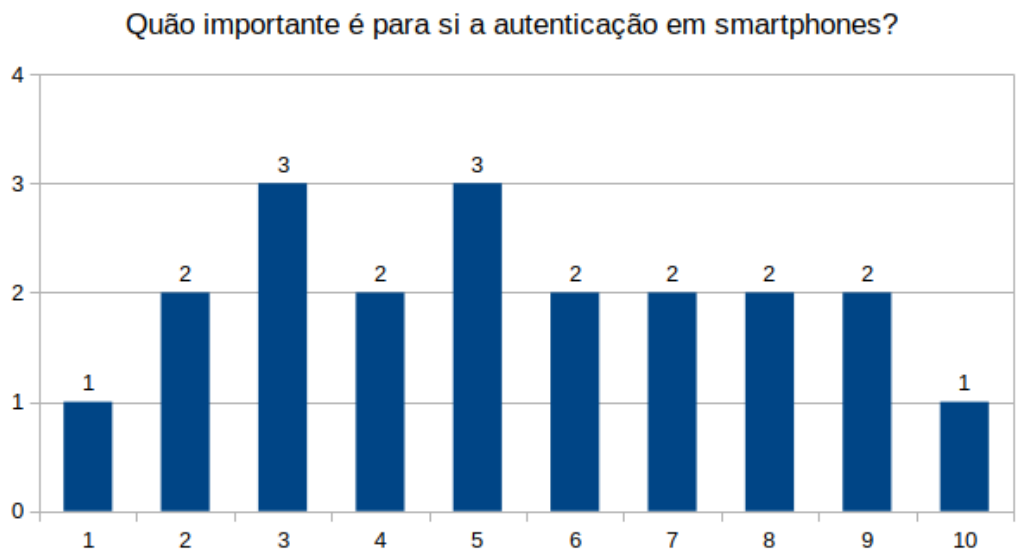


Figura 16: Gráfico das respostas à pergunta 1

Figura 17: Gráfico das respostas à pergunta 2

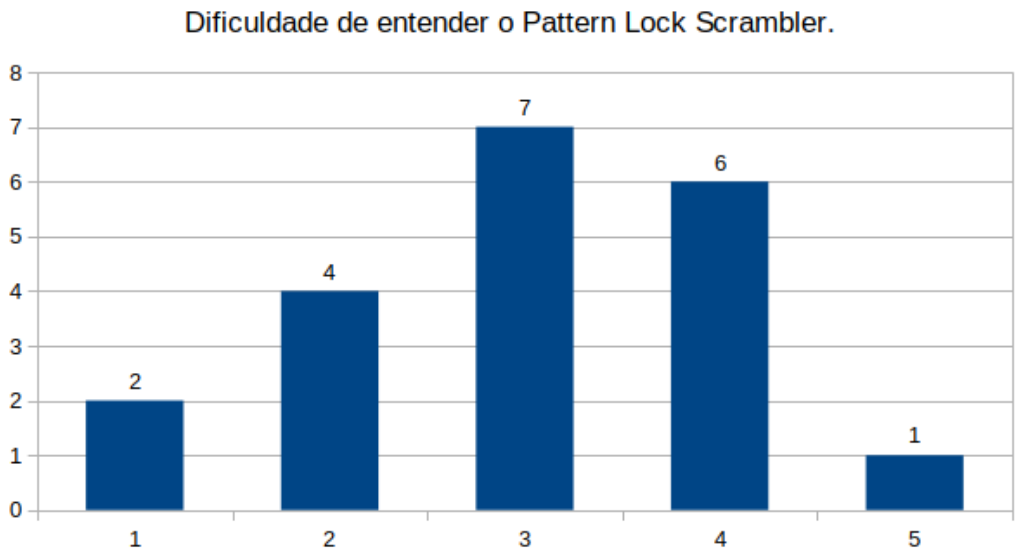


Figura 18: Gráfico das respostas à pergunta 3

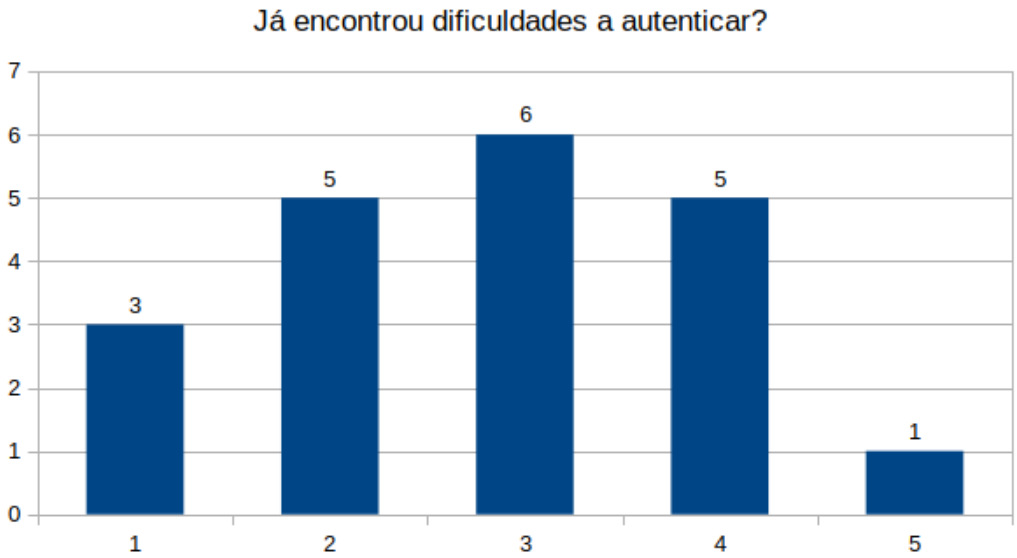


Figura 19: Gráfico das respostas à pergunta 4

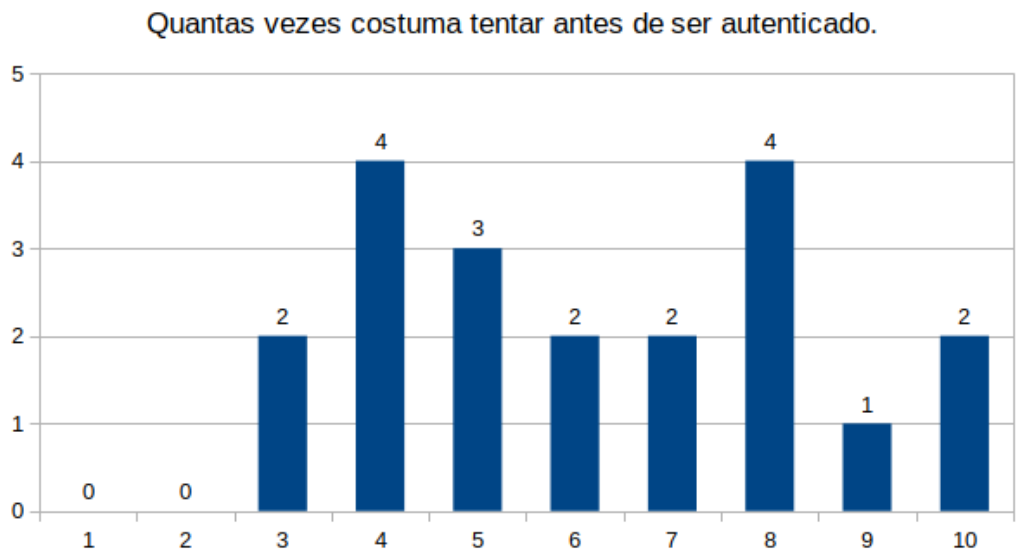


Figura 20: Gráfico das respostas à pergunta 5

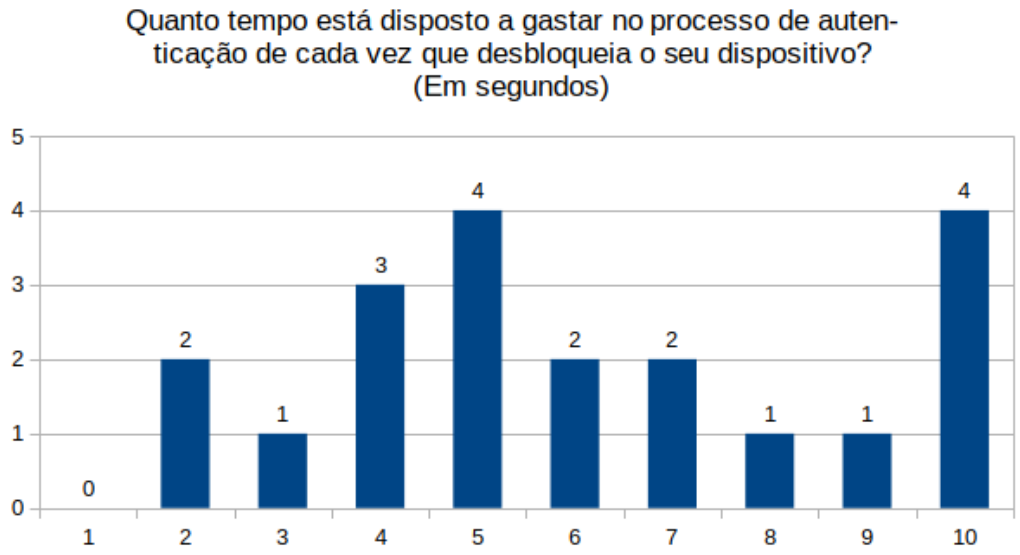


Figura 21: Gráfico das respostas à pergunta 6

Também é tomada nota que enquanto a maioria (70%) percebeu a teórica por trás do mecanismo pelo menos ao nível 3, as tentativas necessárias para autenticar nunca são menos de 3, chegando 2 pessoas a tentar 10 ou mais vezes para autenticar.

A pergunta apresentada na Figura 21 foi incluída para perceber o nível de incómodo que os utilizadores podem tolerar no desbloqueio do dispositivo (ou app). Como se pode verificar pelas respostas, nenhum utilizador respondeu com a opção de 1 segundos e houve uma quantidade de respostas na opção 10 ou mais.

5.3 CONCLUSÃO DOS RESULTADOS

A conclusão imediata retirada dos resultados é que a população de resposta é limitada e como tal não é possível deduzir conclusões de alta confiança. Como a aplicação apresentada aos utilizadores que responderam estava numa fase de produto viável mínimo, era necessário o autor explicar o funcionamento, mesmo que brevemente, para a pessoa poder experimentar a aplicação e assim responder ao inquérito. No estado final da aplicação será bastante reduzida, ou até eliminada, a necessidade de explicar verbalmente o funcionamento da aplicação, pois aí o próprio programa será intuitivo e com instruções.

A não presença de respostas de valores 1 e 2 na pergunta 5 (Figura 20) implica que o funcionamento do método de autenticação não é suficientemente conveniente para o uso diário e o desbloqueio frequente.

6

CONCLUSÕES

Este projeto permitiu-me adquirir conhecimento na área da autenticação, nomeadamente a autenticação *mobile passwordless*, donde extraí informação como, os novos métodos utilizados, as técnicas usadas para validar utilizador, as desvantagens que cada um apresenta, e a importância de haver um sistema robusto de autenticação nos telemóveis que agora fazem parte do dia a dia. O desenvolvimento do projeto também concluiu com uma prova de conceito que considero útil explorar para combater o problema do *shoulder-surfing* ou OTS.

Em suma, o objetivo deste projeto foi investigar os mecanismos de autenticação existentes, sobretudo no âmbito de *smartphones*, e analisar as fraquezas existentes. Das quais fraquezas, foi investigada principalmente uma presente num método de autenticação bastante utilizado em *smartphones*, o bloqueio por padrão, e seguidamente, hipotetizada uma possível mitigação da mesma. Em seguida, elaborou-se um esquema geral de aplicação real da mitigação, que foi aplicada a um caso prático em forma de uma aplicação *standalone* que simula um ecrã de desbloqueio.

Foi construído um inquérito apresentado ao público que após respondido foi analisado para deduzir e extrair conclusões práticas do estado do sistema.

Concluindo a análise de respostas, foi anotado que a aplicação deste método de autenticação seria mais adequado em aplicações menos utilizadas que precisam de mais segurança do que no quotidiano.

6.1 CONTRIBUIÇÕES PARA A ÁREA

Neste projeto foi desenvolvido um método original que visa mitigar as capacidades de ataques OTS. Este método foi desenvolvido numa plataforma *mobile*, para o sistema operativo Android e em Kotlin.

O método foi publicado num repositório *online* com a licença MIT para disponibilizar para o mundo de maneira que outros possam basear ou modificar o código da forma que acharem mais correto.

6.2 RECOMENDAÇÕES PARA A INVESTIGAÇÃO FUTURA

A recomendação mais importante é a continuação do desenvolvimento da aplicação para ser o mais intuitiva possível ao utilizador autorizado e para ter uma adoção do público geral. Com desenvolvimento constante deste método, um dia este poderá ser transformado numa biblioteca pública que programadores podem escolher incluir nos seus projetos de uma maneira direta e sem complicações.

A continuação do desenvolvimento deste método contará com novos passos de transformação do padrão, por exemplo, inverter os pontos, substituir apenas um ponto por outro, definir vários padrões de autenticação diferentes que seguem uma *seed* para definir qual é o de autenticação válida no momento, em vez de haver só um padrão transformado pela seed, entre outras. Também é possível alterar a forma com que os utilizadores podem perceber a *seed* (p.e. hora e minuto). Em vez de ser olharem para o relógio do telemóvel, poderia ser implementado um esquema de cores cujo *background* do padrão (ou *foreground* , ou até uma imagem escolhida pelo utilizador. . .) altera conforme o padrão a ser introduzido. Até seria considerável um esquema no qual o utilizador apenas tem de perceber uma dica em forma de padrão que aparece já selecionado no ecrã, para induzir um atacante em erro, em que assim que o utilizador pressiona um nó desaparece.

No que consta a *seed* escolhida (hora e minuto), é possível alterar esta mesma para que passe a ser outra que não a data/hora. É implementável uma *seed* distinta, visto que o código de transformação não está estaticamente programado para a *timestamp* , logo, uma *seed* como percentagem de bateria, cor de uma imagem (que funciona tanto como dica para outras *seeds* como a *seed* em si diretamente), entre outras.

6.3 LIMITAÇÕES DO ESTUDO

A população que respondeu ao inquérito disponibilizado é reduzida, e como tal, é uma limitação do estudo. A continuação do desenvolvimento da aplicação *mobile* é algo que, por sua eventualidade, reduziria essa limitação.

O produto, como é apresentado num estado mínimo viável para apresentação, contribui fortemente para as limitações do projeto.

O método PatternLock Scrambler depende fortemente de que o atacante não saiba qual é o mecanismo de aleatorização utilizado pela vítima, se souber que

usa o método sequer. Logo, um atacante que tome conhecimento da utilização do método pela vítima poderá, com tentativas suficientes, sempre encontrar o padrão correspondente. O mecanismo só procura chegar ao ponto em que o atacante nunca saiba se um padrão que tentou anteriormente seja, por meio da natureza cíclica do sistema, novamente o padrão real de autenticação.

BIBLIOGRAFIA

[1] *Mobile Operating System Market Share Worldwide | Statcounter Global Stats*, [Online; accessed 27. May 2023], mai. de 2023. URL: https://gs.statcounter.com/os-market-share/mobile/worldwide/#monthly-201101-202304.

[2] Wikipedia contributors, *Android (operating system) — Wikipedia, The Free Encyclopedia*, [Online; accessed 20-May-2023], 2023. URL: https://en.wikipedia.org/w/index.php?title=Android_(operating_system)&oldid=1155708564.

[3] S. G. Lyastani, M. Schilling, M. Neumayr, M. Backes e S. Bugiel, «Is FIDO2 the Kingslayer of User Authentication? A Comparative Usability Study of FIDO2 Passwordless Authentication.», em *IEEE Symposium on Security and Privacy*, 2020, pp. 268–285.

[4] V. Parmar, H. A. Sanghvi, R. H. Patel e A. S. Pandya, «A Comprehensive Study on Passwordless Authentication», em *2022 International Conference on Sustainable Computing and Data Communication Systems (ICSCDS)*, abr. de 2022, pp. 1266–1275. DOI: 10.1109/ICSCDS53736.2022.9760934.

[5] C. Herley, P. v. Oorschot e A. S. Patrick, «Passwords: If We're So Smart, Why Are We Still Using Them?», *Proc. Financial Crypto*, jan. de 2009. URL: https://www.microsoft.com/en-us/research/publication/passwords-if-were-so-smart-why-are-we-still-using-them/.

[6] J. Bonneau, C. Herley, P. C. van Oorschot e F. Stajano, «The Quest to Replace Passwords: A Framework for Comparative Evaluation of Web Authentication Schemes», *2012 IEEE Symposium on Security and Privacy*, pp. 553–567, 2012. URL: https://api.semanticscholar.org/CorpusID:7847705.

[7] N. K. Ratha, J. H. Connell e R. M. Bolle, «Enhancing security and privacy in biometrics-based authentication systems», *IBM Systems Journal*, vol. 40, n.º 3, pp. 614–634, 2001. DOI: 10.1147/sj.403.0614.

[8] *Smartphone sales worldwide 2007-2021 | Statista*, [Online; accessed 14. May 2023], mai. de 2023. URL: https://www.statista.com/statistics/263437/global-smartphone-sales-to-end-users-since-2007.

[9] *Mobile network subscriptions worldwide 2028 | Statista*, [Online; accessed 16. May 2023], mai. de 2023. URL: https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide.

[10] *Screen lock methods for global mobile users 2021 | Statista*, [Online; accessed 16. May 2023], mai. de 2023. URL: https://www.statista.com/statistics/1305993/screen-lock-methods-global-users.

[11] G. Ye, Z. Tang, D. Fang et al., «Cracking Android Pattern Lock in Five Attempts», em *Proceedings 2017 Network and Distributed System Security Symposium*, sér. NDSS '17, San Diego, California, USA: Internet Society, fev. de 2017. DOI: 10.14722/ndss.2017.23130.

[12] A. Cain e J. Still, «Usability Comparison of Over-the-Shoulder Attack Resistant Authentication Schemes - JUX», *JUX - The Journal of User Experience*, ago. de 2018. URL: https://uxpajournal.org/usability-osa-resistant-authentication.

[13] T. Stockinger, «Implicit authentication on mobile devices», vol. 8, 2011.

[14] *What is FIDO? - FIDO Alliance*, [Online; accessed 22. Apr. 2023], mar. de 2023. URL: https://fidoalliance.org/what-is-fido.

[15] F. Aldhaban, «Exploring the adoption of Smartphone technology: Literature review», em *2012 Proceedings of PICMET '12: Technology Management for Emerging Technologies*, jul. de 2012, pp. 2758–2770.

[16] C. Bröhl, P. Rasche, J. Jablonski, S. Theis, M. Wille e A. Mertens, «Desktop PC, Tablet PC, or Smartphone? An Analysis of Use Preferences in Daily Activities for Different Technology Generations of a Worldwide Sample», em *Human Aspects of IT for the Aged Population. Acceptance, Communication and Participation*, Springer International Publishing, 2018, pp. 3–20. DOI: 10.1007/978-3-319-92034-4_1.

[17] *Desktop vs Mobile vs Tablet Market Share Worldwide | Statcounter Global Stats*, [Online; accessed 1. Aug. 2023], set. de 2023. URL: https://gs.statcounter.com/platform-market-share/desktop-mobile-tablet/worldwide/#monthly-201612-201712.

[18] S. Malviya, «The Triad of Functionality, Usability, and Security», *Medium*, jun. de 2023. URL: https://medium.com/@sumitmalviyaofc/the-triad-of-functionality-usability-and-security-17c6b651235f.

[19] *Kotlin for Android | Kotlin*, [Online; accessed 20. May 2023], mai. de 2023. URL: https://kotlinlang.org/docs/android-overview.html.

[20] F. Muntenescu, «Fewer crashes and more stability with Kotlin - Android Developers - Medium», *Medium*, dez. de 2021. URL: https://medium.com/androiddevelopers/fewer-crashes-and-more-stability-with-kotlin-b606c6a6ac04.

[21] M. Jakobsson, E. Shi, P. Golle, R. Chow et al., «Implicit authentication for mobile devices», em *Proceedings of the 4th USENIX conference on Hot topics in security*, USENIX Association, vol. 1, 2009, pp. 25–27.

[22] E. von Zezschwitz, P. Dunphy e A. D. Luca, «Patterns in the wild», em *Proceedings of the 15th international conference on Human-computer interaction with mobile devices and services*, ACM, ago. de 2013. DOI: 10.1145/2493190.2493231.

[23] A. D. Luca, A. Hang, F. Brudy, C. Lindner e H. Hussmann, «Touch me once and i know it's you!», em *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, mai. de 2012. DOI: 10.1145/2207676.2208544.

[24] M. Tamviruzzaman, S. I. Ahamed, C. S. Hasan e C. O'brien, «ePet When Cellular Phone Learns to Recognize Its Owner», em *Proceedings of the 2nd ACM workshop on Assurable and usable security configuration*, ACM, nov. de 2009. DOI: 10.1145/1655062.1655066.

[25] N. L. Clarke e S. M. Furnell, «Authenticating mobile phone users using keystroke analysis», *International Journal of Information Security*, vol. 6, n.º 1, pp. 1–14, ago. de 2006. DOI: 10.1007/s10207-006-0006-6.

[26] S. Yazji, X. Chen, R. P. Dick e P. Scheuermann, «Implicit User Re-authentication for Mobile Devices», em *Ubiquitous Intelligence and Computing*, Springer Berlin Heidelberg, 2009, pp. 325–339. DOI: 10.1007/978-3-642-02830-4_25.

[27] O. Riva, C. Qin, K. Strauss e D. Lymberopoulos, «Progressive Authentication: Deciding When to Authenticate on Mobile Phones.», em *USENIX Security Symposium*, 2012, pp. 301–316.

[28] M. Harbach, E. von Zezschwitz, A. Fichtner, A. D. Luca e M. Smith, «It's a Hard Lock Life: A Field Study of Smartphone (Un)Locking Behavior and Risk Perception», em *10th Symposium On Usable Privacy and Security (SOUPS 2014)*, Menlo Park, CA: USENIX Association, jul. de 2014, pp. 213–230, ISBN: 978-1-931971-13-3. URL: https://www.usenix.org/conference/soups2014/proceedings/presentation/harbach.

[29] V. Beal, «What is Authentication?», *Webopedia*, jun. de 2022. URL: https://www.webopedia.com/definitions/authentication.

[30] S. Gupta, A. Buriro e B. Crispo, «Demystifying Authentication Concepts in Smartphones: Ways and Types to Secure Access», *Mobile Information Systems*, vol. 2018, pp. 1–16, 2018. DOI: 10.1155/2018/2649598.

[31] N. A. Lal, S. Prasad e M. Farik, «A review of authentication methods», *vol*, vol. 5, pp. 246–249, 2016.

[32] G. Kambourakis, D. Damopoulos, D. Papamartzivanos e E. Pavlidakis, «Introducing touchstroke: keystroke-based authentication system for smartphones», *Security and Communication Networks*, vol. 9, n.º 6, pp. 542–554, jul. de 2014. DOI: 10.1002/sec.1061.

[33] A. Alzubaidi e J. Kalita, «Authentication of Smartphone Users Using Behavioral Biometrics», *IEEE Communications Surveys & Tutorials*, vol. 18, n.º 3, pp. 1998–2026, 2016. DOI: 10.1109/comst.2016.2537748.

[34] *Which password authentication method works best for businesses?*, [Online; accessed 28. Feb. 2023], mar. de 2021. URL: https://www.passportalmsp.com/blog/which-password-authentication-method-works-best-businesses.

[35] Y. Li, H. Hu e G. Zhou, «Using Data Augmentation in Continuous Authentication on Smartphones», *IEEE Internet of Things Journal*, vol. 6, n.º 1, pp. 628–640, fev. de 2019. DOI: 10.1109/jiot.2018.2851185.

[36] A. Buriro, B. Crispo, F. D. Frari, J. Klardie e K. Wrona, «ITSME: Multimodal and Unobtrusive Behavioural User Authentication for Smartphones», em *Technology and Practice of Passwords*, Springer International Publishing, 2016, pp. 45–61. DOI: 10.1007/978-3-319-29938-9_4.

[37] M. Faundez-Zanuy, «Biometric security technology», *IEEE Aerospace and Electronic Systems Magazine*, vol. 21, n.º 6, pp. 15–26, jun. de 2006. DOI: 10.1109/maes.2006.1662038.

[38] L. Yang, Y. Guo, X. Ding et al., «Unlocking Smart Phone through Handwaving Biometrics», English, *IEEE Transactions on Mobile Computing*, vol. 14, n.º 5, pp. 1044–1055, mai. de 2015, ISSN: 1536-1233. DOI: 10.1109/TMC.2014.2341633.

[39] B. Shrestha, N. Saxena e J. Harrison, «Wave-to-Access: Protecting Sensitive Mobile Device Services via a Hand Waving Gesture», em *Cryptology and Network Security*, Springer International Publishing, 2013, pp. 199–217. DOI: 10.1007/978-3-319-02937-5_11.

[40] A. D. Luca e J. Lindqvist, «Is secure and usable smartphone authentication asking too much?», *Computer*, vol. 48, n.º 5, pp. 64–68, mai. de 2015. DOI: 10.1109/mc.2015.134.

[41] A. D. Luca, A. Hang, E. von Zezschwitz e H. Hussmann, «I Feel Like I'm Taking Selfies All Day!», em *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, ACM, abr. de 2015. DOI: 10.1145/2702123.2702141.

[42] C. Bhagavatula, B. Ur, K. Iacovino, S. M. Kywe, L. F. Cranor e M. Savvides, «Biometric Authentication on iPhone and Android: Usability, Perceptions, and Influences on Adoption», em *Proceedings 2015 Workshop on Usable Security*, Internet Society, 2015. DOI: 10.14722/usec.2015.23003.

[43] A. K. Jain, *Biometrics Personal Identification In Networked Society, Personal Identification In Networked Society*. Springer, 2013, p. 424, ISBN: 9781475782950.

[44] *Unlock your Pixel phone with your face - Pixel Phone Help*, [Online; accessed 20. Jun. 2023], jul. de 2023. URL: https://support.google.com/pixelphone/answer/9517039?hl=en&ref_topic=7083614&sjid=14470493004732746897-EU.

[45] *O que é Smart Lock no Android [e como isso funciona] – Tecnoblog*, [Online; accessed 1. Mar. 2023], fev. de 2023. URL: https://tecnoblog.net/responde/o-que-e-smart-lock-no-android.

[46] M. M. Koushki, B. Obada-Obieh, J. H. Huh e K. Beznosov, «Is Implicit Authentication on Smartphones Really Popular? On Android Users' Perception of Smart Lock for Android», em *22nd International Conference on Human-Computer Interaction with Mobile Devices and Services*, ACM, out. de 2020. DOI: 10.1145/3379503.3403544.

[47] F. Morgner, D. Oepen, W. Müller e J.-P. Redlich, «Mobile Smart Card Reader Using NFC-Enabled Smartphones», em *Security and Privacy in Mobile Information and Communication Systems*, Springer Berlin Heidelberg, 2012, pp. 24–37. DOI: 10.1007/978-3-642-33392-7_3.

[48] *Choose when your Pixel phone can stay unlocked - Pixel Phone Help*, [Online; accessed 2. Mar. 2023], mar. de 2023. URL: https://support.google.com/pixelphone/answer/6093922?hl=en#zippy=%5C%2Ckeep-your-phone-unlocked-when-its-at-a-trusted-place.

[49] Y. Song, G. Cho, S. Oh, H. Kim e J. H. Huh, «On the Effectiveness of Pattern Lock Strength Meters», em *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, ACM, abr. de 2015. DOI: 10.1145/2702123.2702365.

[50] L. Zhang, C. Bo, J. Hou et al., «Kaleido», em *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, ACM, set. de 2015. DOI: 10.1145/2789168.2790106.

[51] *NIST Special Publication 800-63-3*, [Online; accessed 1. May 2023], mai. de 2023. URL: https://pages.nist.gov/800-63-3/sp800-63-3.html#sec3.

[52] *NIST Special Publication 800-63B*, [Online; accessed 2. May 2023], mai. de 2023. URL: https://pages.nist.gov/800-63-3/sp800-63b.html#multifactorOTP.

[53] M. Casey, M. Manulis, C. J. P. Newton, R. Savage e H. Treharne, «An Interoperable Architecture for Usable Password-Less Authentication», em *Emerging Technologies for Authorization and Authentication*, A. Saracino e P. Mori, eds., Cham: Springer International Publishing, 2020, pp. 16–32, ISBN: 978-3-030-64455-0.

[54] *What Is Multi-Factor Authentication?*, [Online; accessed 2. May 2023], abr. de 2023. URL: https://www.cisco.com/c/en/us/products/security/what-is-multi-factor-authentication.html#~how-mfa-works.

[55] B. Lutkevich, «cryptographic nonce», *Security*, out. de 2021. URL: https://www.techtarget.com/searchsecurity/definition/nonce.

[56] *OAuth 2.0 — OAuth*, [Online; accessed 2. May 2023], mai. de 2023. URL: https://oauth.net/2.

[57] *RFC ft-ietf-oauth-v2: The OAuth 2.0 Authorization Framework*, [Online; accessed 2. May 2023], out. de 2012. URL: https://datatracker.ietf.org/doc/html/rfc6749.

[58] *IBM Documentation*, [Online; accessed 2. May 2023], mar. de 2021. URL: https://www.ibm.com/docs/en/ztpf/1.1.0.15?topic=authentication-what-is-digital-certificate.

[59] *Overview*, [Online; accessed 14. Jul. 2023], set. de 2023. URL: https://developers.yubico.com/U2F/Protocol_details/Overview.html.

[60] É. C. Pastilha, «Autenticação FIDO2 Em Dispositivos Android», tese de mestrado, abr. de 2022. URL: http://hdl.handle.net/10400.8/7321.

[61] Contributors to Wikimedia projects, *Hash function - Wikipedia*, [Online; accessed 9. May 2023], mar. de 2023. URL: https://en.wikipedia.org/w/index.php?title=Hash_function&oldid=1142599950.

[62] P. Christensson, «Hash», *TechTerms.com*, abr. de 2018. URL: https://techterms.com/definition/hash.

[63] *Global mobile OS market share 2022 | Statista*, [Online; accessed 27. May 2023], mai. de 2023. URL: https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009.

[64] B. King, «Is Android Really Open-Source? And Does It Even Matter?», *MUO*, dez. de 2021. URL: https://www.makeuseof.com/tag/android-really-open-source-matter.

[65] *Download Android Studio & App Tools - Android Developers*, [Online; accessed 20. May 2023], mai. de 2023. URL: https://developer.android.com/studio.

[66] *Comparison to Java | Kotlin*, [Online; accessed 15. Aug. 2023], set. de 2023. URL: https://kotlinlang.org/docs/comparison-to-java.html.

[67] *SQLite Home Page*, [Online; accessed 27. Sep. 2023], set. de 2023. URL: https://www.sqlite.org/index.html.

DECLARAÇÃO

Declaro, sob compromisso de honra, que o trabalho apresentado nesta dissertação, com o título *"PatternLockScrambler: Impedir ataques over-the-shoulder ao bloqueio por padrão"*, é original e foi realizado por Ricardo Perpétuo Cação (2213267) sob orientação de Professor Ricardo Jorge Pereira Gomes (ricardo.p.gomes@ipleiria.pt) e Professora Doutora Sónia Maria Almeida da Luz (sonia.luz@ipleiria.pt).

Leiria, Setembro de 2023

Ricardo Perpétuo Cação

Assinado por: **RICARDO PERPÉTUO CAÇÃO**
Num. de Identificação: BI30034090
Data: 2023.09.28 18.30.47 GMT Daylight time

CHAVE MÓVEL