UNIVERSIDADE FEDERAL DO PAMPA

Rafael Fernandes da Silva

S3AS: uma Solução de Autenticação e Autorização através de Aplicativos de Smartphones

Alegrete
2019

Rafael Fernandes da Silva

S3AS: uma Solução de Autenticação e Autorização através de Aplicativos de Smartphones

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação da Universidade Federal do Pampa como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof. Me. Diego Luis Kreutz

Alegrete
2019

Rafael Fernandes da Silva

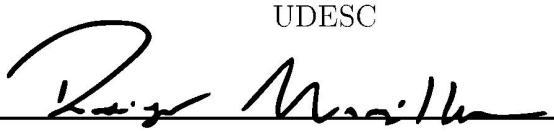
S3AS: uma Solução de Autenticação e Autorização através de Aplicativos de Smartphones

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação da Universidade Federal do Pampa como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

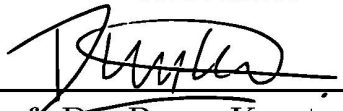
Trabalho de Conclusão de Curso defendido e aprovado em 28 de novembro de 2019
Banca examinadora:

Prof. Me. Diego Luis Kreutz
Orientador
UNIPAMPA

Prof. Dr. Charles Christian Miers
UDESC



Prof. Dr. Rodrigo Brandão Mansilha
UNIPAMPA



Prof. Dr. Roger Kreutz Immich
UFRN

À todas as mulheres da minha vida,
Mãe, Irmãs e Sobrinhas.

AGRADECIMENTOS

Listar todas as pessoas que fizeram parte da minha trajetória é uma tarefa complicada. A fim de evitar esquecimentos, não citarei os nomes de todos que participaram da minha vida, mas a todos que sabem de sua importância, deixo os meus mais sinceros agradecimentos.

Ressalto o carinho, afeto, confiança e tudo que foi proporcionado por todas as mulheres da minha vida, mãe, irmãs e sobrinhas, já que são minha motivação e sem todas elas, eu não estaria realizando meu sonho. Indico, também, o sentimento de gratidão aos meus colegas do estado e também os que vêm de fora por me proporcionarem diversas experiências e valores culturais que levarei para o resto da vida. Saliento a relevância da amizade que mantive com os meus colegas Felipe Melchior, Guilherme Neri e Lucas Antunes, principalmente na fase de conclusão de curso, pois me ajudaram quando a solução me parecia abstrata e me incentivaram a não desistir quando não era possível enxergar uma saída.

Sem sombra de dúvidas, uma das minhas maiores influências e referências como ser humano, ou melhor, como um bom ser humano é o meu pai. Acredito que, onde ele quer que ele esteja, seu coração se enche de orgulho ao ver o que estou me tornando. Antes de finalizar esta dedicatória, não posso deixar de agradecer enormemente ao meu orientador, professor e amigo Diego Kreutz, que aceitou assumir este papel com boa vontade, me ajudando quando necessário, sendo paciente quando fui teimoso e me atendendo sempre com prestatividade.

RESUMO

A crescente utilização de aplicativos, especialmente em dispositivos móveis, como forma de autenticação de usuários está trazendo à tona novas oportunidades e desafios de segurança. Com o objetivo de modernizar suas formas de acesso, algumas instituições estão adotando QR Codes estáticos gerados a partir de informações simples e imutáveis, como o CPF dos seus associados. Esse procedimento possui implementação e verificação fáceis, mas representa uma vulnerabilidade crítica sob a ótica da segurança. Com o objetivo de mitigar essa vulnerabilidade, este trabalho propõe uma Solução de Autenticação e Autorização através de Aplicativos de Smartphones, denominada S3AS. A solução proposta é composta de dois protocolos principais, um de vinculação de credenciais do usuário (i.e., identificação) ao dispositivo móvel e outro para a geração de códigos de autenticação descartáveis (OTACs). Ambos os protocolos foram formalmente verificados utilizando a ferramenta de verificação automática Scyther. Os resultados indicam que os protocolos da S3AS são robustos e seguros segundo as análises automáticas realizadas com a ferramenta Scyther. Como forma de demonstrar o funcionamento e a viabilidade técnica da solução, foi implementado também um protótipo que simula o controle de acesso utilizando catracas eletrônicas. Os números das execuções experimentais indicam que o tempo gasto para geração e utilização de OTACs é baixo. Um OTAC pode ser gerado e verificado em menos de 1 (um) milissegundo utilizando a linguagem de programação Python.

Palavras-chave: Segurança da Informação. Autenticação. OTP.

ABSTRACT

The increasing adoption of mobile applications as a means of user authentication is revealing new security challenges and opportunities. In order to modernize their physical authentication and authorization procedures (e.g., access turnstile), some institutions have been adopted static QR Codes generated using simple and static user data, as the Individual Taxpayer Registration (CPF, in Brazil). This procedure is easy to implement and verify, but it represents a critical security vulnerability. Aiming to mitigate this vulnerability, this paper proposes an Authentication and Authorization Solution based on Smartphone Applications, named S3AS. The proposed solution consists of two main protocols, one for binding user credentials to the mobile device (i.e., identification) and another one for generating one-time authentication codes (OTACs). Both protocols have been formally verified using the automatic verification tool named Scyther. Based on the automated analysis done with the Scyther tool, the results show that the S3AS protocols are robust and safe. A prototype to simulate access control using electronic turnstiles was also developed to demonstrate how the solution works and its deployment viability. The numbers of experimental runs indicate that the cost of using and generating OTACs is quite low. An OTAC can be generated and verified in less than 1 millisecond using the Python programming language.

Key-words: Information Security. Authentication. OTP.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

AES Advanced Encryption Standard

CPF Cadastro de Pessoa Física

FPGA Field Programmable Gate Array

HMAC Hash-based Message Authentication

IMEI International Mobile Equipment Identity

NFC Near-Field Communication

OTAC One-Time Authentication Code

PFS Perfect Forward Secrecy

PKE Public Key Encryption

QR Code Quick Response Code

RFID Radio-Frequency Ientification

S3AS Solução de Autenticação e Autorização através de Aplicativos de Smartphones

SESC-RS Serviço Social do Comércio do Rio Grande do Sul

SHA Secure Hashing Algorithm

SMS Short Message Service

TAE Técnico-administrativo em educação

TCP/IP Transmission Control Protocol/Internet Protocol

TLS Transport Layer Security

SUMÁRIO

1 INTRODUÇÃO

Os dispositivos móveis já fazem parte do cotidiano da maioria das pessoas. Com o advento da Internet das Coisas e serviços disponíveis em servidores na Nuvem, este cenário deve se tornar ainda mais comum nos próximos anos (CURADO et al., 2019; BITTENCOURT et al., 2018). Segundo estatísticas, atualmente o Brasil possui mais de 420 milhões de dispositivos móveis conectados à Internet. Desse total, cerca de 235 milhões são *smartphones*, o que representa mais de um aparelho desse tipo habilitado por habitante (FGV, 2019).

Uma das utilizações dos *smartphones*, ainda que incipiente, é para a autenticação (*i.e.*, identificação e verificação) para controle de acesso (permissão ou negação de entrada) a um espaço (físico ou lógico/virtual). Por exemplo, *smartphones* estão sendo utilizados para identificar e permitir o acesso de usuários às instalações do Serviço Social do Comércio do Rio Grande do Sul (SESC-RS)[1] através do cartão virtual de usuário (SESC/RS, 2018). Entretanto, o cartão virtual contém um código de autenticação que é apenas um Quick Response Code (QR Code) estático contendo o número do Cadastro de Pessoa Física (CPF) acrescido de um dígito (geralmente "0"). Mecanismos de autenticação como esse são conhecidos como de autenticação de único fator estático. O único fator estático é relativamente simples de ser burlado, pois basta ao agente malicioso obter acesso às credenciais do usuário para comprometer o mecanismo de autenticação. Por exemplo, no caso do SESC-RS, que utiliza um único fator estático, o agente malicioso precisa apenas conhecer o CPF da pessoa ou clonar o QR Code. É importante observar que um QR Code pode ser facilmente lido à distância devido a sua própria natureza e propósito e que isso, porém, facilita sua clonagem. Segundo pesquisas recentes, vazamento de credenciais de usuários é o meio responsável pela maioria dos acessos indevidos a dados (MACHADO et al., 2019a).

Buscando mitigar algumas vulnerabilidades de segurança, foram propostos protocolos de autenticação de múltiplos fatores (PIETRO; ME; STRANGIO, 2005; MALIKI; SEIGNEUR, 2007; Starnberger; Froihofer; Goeschka, 2009; LEE et al., 2010; KAUR; DEVGAN; BHUSHAN, 2016; FERRAG et al., 2018; Wu et al., 2019). Uma autenticação de dois (ou mais) fatores pode ser realizada através da utilização de usuário/senha como o primeiro fator e um código de autenticação, gerado por uma aplicação específica (*e.g.*, Google Authenticator) ou enviado via Short Message Service (SMS), como segundo fator. Entretanto, a autenticação de múltiplos fatores leva a questões e desafios de usabilidade (CRISTOFARO et al., 2013). Tomando como o exemplo o cartão virtual do SESC-RS, os usuários desejam passar rapidamente pelas catracas para acessar as dependências da instituição, como as academias, quadras poliesportivas, ginásios, parque aquático, entre outros. Nesse exemplo, um fator adicional de autenticação pode aumentar o tempo de transito pelas catracas e reduzir a usabilidade da solução. A principal questão

[1] <https://www.sesc-rs.com.br>

que surge é: *como resolver o problema da autenticação, utilizando um único fator, sem comprometer a segurança e a usabilidade?*

O objetivo deste trabalho é propor uma Solução de Autenticação e Autorização através de Aplicativos de Smartphones (S3AS) (FERNANDES et al., 2019a; FERNANDES et al., 2019b) A principal finalidade da S3AS é prover autenticação segura, de um único fator dinâmico, sem comprometer a usabilidade. A solução é composta essencialmente por dois protocolos, um de vinculação de credenciais do usuário (*i.e.*, identificação) ao dispositivo móvel e outro de geração de códigos de autenticação descartáveis, denominados One-Time Authentication Codes (sOTACs). A ideia principal é limitar a vinculação da identidade do usuário a apenas um único *smartphone*, reduzindo a possibilidade de utilização das mesmas credenciais por múltiplas pessoas em múltiplos dispositivos. O protocolo de vinculação gera uma chave mestra, que é utilizada pelo segundo protocolo para derivar códigos de autenticação únicos.

Este trabalho apresenta como principais contribuições: (1) projeto e discussão de um protocolo de vinculação de usuários a dispositivos; (2) o projeto conceitual e desenvolvimento de um protocolo de geração de códigos de autenticação únicos; (3) discussão sobre casos de uso para demonstrar a aplicação prática da solução; (4) verificação formal dos protocolos da solução utilizando a ferramenta Scyther; e (5) implementação de um protótipo da S3AS que simula a utilização da solução em catracas eletrônicas através de QR Code.

O restante deste trabalho está organizado conforme descrito a seguir. No Capítulo 2 são apresentados e discutidos os requisitos, os pressupostos e os protocolos da solução proposta. A verificação formal dos protocolos da S3AS e uma avaliação experimental são apresentadas nos Capítulos 3 e 4. Por fim, as discussões, os trabalhos relacionados e a conclusão aparecem nos Capítulos 5, 6 e 7, respectivamente.

2 DESENVOLVIMENTO

Este capítulo apresenta o desenvolvimento da solução S3AS, esta que é composta por dois protocolos: identificação, em que o aplicativo do usuário é vinculado a um único dispositivo móvel, e verificação, no qual são gerados códigos descartáveis de autenticação. Na Seção 2.1 são apresentados os requisitos e pressupostos da solução. As Seções 2.2 e 2.3 descrevem os protocolos de identificação e de verificação, respectivamente. Por fim, na Seção 2.4 é apresentado a solução aplicada a quatro casos de uso distintos.

2.1 Requisitos e Pressupostos

Os requisitos descrevem as propriedades necessárias que devem ser atendidas no projeto e implementação da solução. Eis os principais requisitos da solução.

Vínculo único. Um usuário deve estar vinculado a um único dispositivo em um dado instante de tempo t. O vínculo único reduz os riscos associados a eventual tentativa de utilização de credencias de usuários vazadas, por exemplo. No caso de um vazamento das credenciais do usuário, o que é algo comum e frequente no contexto atual (MACHADO et al., 2019b), um atacante não deve conseguir vincular o aplicativo, em outro *smartphone* qualquer, utilizando as credenciais do usuário.

Solução de propósito geral. A solução proposta deve ser de propósito geral, ou seja, deve ser aplicável a diferentes cenários de identificação e verificação de usuários. Exemplos de cenários incluem controle de acesso através de catracas eletrônicas, sistemas de ponto eletrônico, sistemas de gerenciamento de chaves de salas e controle de acesso em portões eletrônicos.

Canais *out-of-band* são utilizados para mitigar problemas de interceptação de dados. Enquanto que os sistemas que utilizam um único canal de comunicação facilitam o trabalho dos atacantes, os sistemas baseados em múltiplos fatores de autenticação utilizam canais *out-of-band* para enviar códigos de verificação, dificultando significativamente o trabalho dos atacantes. Estudos mostram que o risco de comprometimento de contas de usuários, em sistemas *on-line*, pode reduzir em até 99.9% com a utilização de múltiplos fatores e canais de autenticação (ZDNET, 2019). Para garantir um vínculo único e seguro, a solução proposta deve utilizar canais *out-of-band* para enviar códigos adicionais de segurança. Estes códigos são comuns em sistemas que exigem altos níveis de segurança e realizam vínculo das aplicações aos dispositivos móveis, como é o caso de bancos como o Revolut[1] e o N26[2].

Revogação de identidades comprometidas. Para casos de roubo ou perda do *smartphone* do usuário, a solução proposta deve oferecer mecanismos para identificar e revogar o vínculo do aplicativo do dispositivo. Os mecanismos devem ser, preferencialmente, de ação automática e autônoma.

[1] <https://www.revolut.com>
[2] <https://n26.com>

Independência de tecnologia. A solução não deve limitar-se a uma única tecnologia ou formato de dados (*e.g.*, QR Code). A solução deve ser utilizável com tecnologias como leitores de QR Code, leitores de códigos de barras, leitores Near-Field Communication (NFC)[3] e sistemas de comunicação via Bluetooth[4].

Os pressupostos são condições assumidas como garantidas no contexto da solução proposta.

Primitivas e protocolos de segurança. É assumido que primitivas como funções *hash* criptográficas (*e.g.*, Secure Hashing Algorithm (SHA)2 e SHA3[5]) e funções de assinatura de mensagens Hash-based Message Authentication (HMAC)[6] (*e.g.,* HMAC-SHA256[7]) são seguras. De forma similar, algoritmos de criptografia simétrica, como o Advanced Encryption Standard (AES)[8], e protocolos como o Transport Layer Security (TLS) v1.3[9] são assumidos como confiáveis e seguros, i.e., são capazes de assegurar propriedades básicas de segurança como confidencialidade de dados armazenados ou em trânsito.

Segurança adicional dos canais *out-of-band*. O número de canais *out-of-band* impacta a segurança de sistemas de forma significativa. Por exemplo, um estudo recente da Microsoft mostra que múltiplos fatores de autenticação, utilizando múltiplos canais, pode bloquear em até 99,9% os ataques automatizados (ZDNET, 2019). A Google também obteve sucesso ao adicionar um canal *out-of-band* (GOOGLE, 2019), como um número de telefone, para recuperar as contas em seu sistema, conseguindo bloquear cerca de 99% dos ataques automatizados, além de reduzir o impacto de ataques como os de *phishing*.

Levando em consideração os dados apresentados pela Microsoft, o gráfico da Figura 1 ilustra o risco de incidentes de segurança (*e.g.*, acesso não autorizado realizado por um agente malicioso) em sistemas de autenticação de acordo com a quantidade de canais. Como pode ser observado, o risco tende rapidamente a zero. A cada canal adicionado no sistema, o risco é reduzido em 99,9%. Em um sistema com 0 canais adicionais, o risco de segurança é 1. A adição de 1 canal faz com que o risco caia para 0,01 e ao utilizar 2 canais, o risco cai para 0,000001. Observa-se que o aumento de canais faz com que o risco de segurança tenda rapidamente a zero.

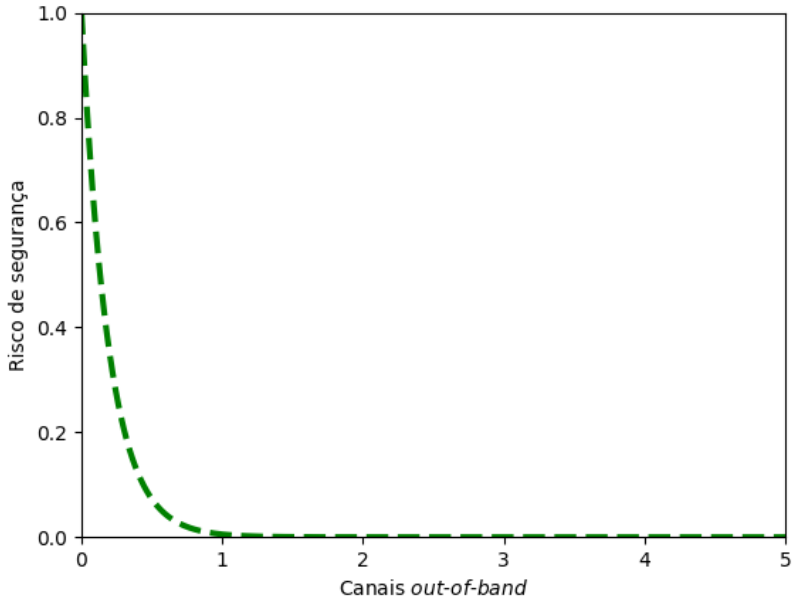
A adição de canais *out-of-band* em um sistema deve levar em consideração o custo para implantá-lo e qual é o nível de segurança necessário para a aplicação. Tomando como exemplo a utilização de um canal de SMS, os preços encontrados para envio de SMS para dispositivos no Brasil são de R$0,08 (SENDPULSE, 2019), R$0,09 (TOTALVOICE, 2019) e R$0,11 (GOI, 2019), respectivamente. Considerando o valor mais baixo, o custo para enviar 50 mil SMSs seria de 4 mil reais. Vale ressaltar que os preços para o envio de SMS

[3] <https://techterms.com/definition/nfc>
[4] <https://techterms.com/definition/bluetooth>
[5] <https://csrc.nist.gov/projects/hash-functions>
[6] <https://csrc.nist.gov/publications/detail/fips/198/1/final>
[7] <https://tools.ietf.org/html/rfc4634>
[8] <https://www.nist.gov/publications/advanced-encryption-standard-aes>
[9] <https://tools.ietf.org/html/rfc8446>

podem variar de acordo com cada país.

Há outras formas de aplicação de um canal *out-of-band* como BB Code utilizado na aplicação do Banco do Brasil (Banco do Brasil, 2019). A solução é um leitor de QR Code de autorização de transações financeiras, disponível no dispositivo móvel do cliente do banco. Para cada transação financeira realizada no sistema *on-line* (*e.g.*, via navegador), é gerado um QR Code único de autorização. Este QR Code contém os dados da transação e é lido e interpretado pelo BB Code. O resultado é um código de autorização da transação, que deve ser digitado pelo cliente no sistema *on-line*. Sem o código de autorização, a transação financeira não é realizada. Neste exemplo, o QR Code contém informações únicas das transações, que são utilizadas para gerar um código único de autorização.

Figura 1: Risco de segurança versus número de canais.



Fonte: Elaborado pelo autor.

***Smartphone* do usuário.** Assume-se que o dispositivo do usuário não é confiável, ou seja, pode ser comprometido. Ao comprometer o dispositivo, o atacante pode ter acesso aos dados de identificação e verificação. Para evitar que o atacante faça uso destes dados, é necessário detectar que o *smartphone* foi comprometido, revogar as credenciais do usuário no aplicativo que utiliza os mecanismos de identificação e verificação, e remover e instalar novamente o aplicativo.

Limite de tentativas de autenticação. O número de tentativas de verificação da identidade e credenciais de acesso do usuário deve ser limitado. Considerando um dispositivo comprometido, o atacante pode utilizar as credenciais do usuário para acessar ambientes, sistemas ou ainda realizar ataques de saturação de recursos. O limite de ten-

tativas de autenticação impede o progresso de ataques de saturação de recursos baseados em infinitas autenticações, por exemplo.

2.2 Protocolo de identificação da aplicação

Após a instalação do aplicativo, o processo inicia com alguma forma de identificação e autenticação do usuário, como por exemplo, *login* e senha. Na sequência, durante o primeiro acesso, é iniciado o protocolo de registro e vinculação do aplicativo. Na solução S3AS, um único dispositivo pode ser vinculado ao fator de conhecimento (*e.g.*, login/senha) do usuário. O protocolo de identificação faz com que, em caso de uma eventual clonagem de dispositivo, a ação seja detectada de forma automática. O passo a passo detalhando este procedimento é apresentado no Tabela 1.

Tabela 1: Registro da aplicação e geração da chave mestra.

1. Cliente $\overset{\text{TLS}}{\longleftrightarrow}$ Servidor	Conexão com verificação do certificado do Servidor
2. Servidor $\rightarrow$ Cliente	[CODE_TLS, $code_1$]
3. Servidor $\rightarrow$ Cliente	[CODE_SMS, $code_2$]
4. Servidor $\rightarrow$ Cliente	[CODE_EMAIL, $code_3$]
5. Cliente, Servidor	$K_{T1} \leftarrow \text{H}(tls\_session\_key \| \| code_1 \| \| code_2 \| \| code_3)$
6. Cliente $\rightarrow$ Servidor	[Cliente, *nonce*, $\text{E}_{K_{T1}}$(IMEI, app_rand1)], $\text{HMAC}_{K_{T1}}$
7. Cliente, Servidor	$K_{T2} \leftarrow \text{H}(IMEI \| \| app\_rand1 \| \| K_{T1})$
8. Servidor $\rightarrow$ Cliente	[Servidor, *nonce*, $\text{E}_{K_{T2}}(server\_rand)$], $\text{HMAC}_{K_{T2}}$
9. Cliente, Servidor	$K_m \leftarrow \text{H}(K_{T1} \| \| K_{T2} \| \| IMEI \| \| app\_rand1 \| \| server\_rand)$
10. Cliente $\rightarrow$ Servidor	[Cliente, V_MKEY, *nonce*, $\text{E}_{K_m}(app\_rand2)$], HMAC_{K_m}
11. Servidor $\rightarrow$ Cliente	[Servidor, V_MKEY, *nonce*, $\text{E}_{K_m}(app\_rand2 + 1)$], HMAC_{K_m}

Fonte: Elaborado pelo autor

O processo de registro inicia com uma conexão utilizando o TLS entre o aplicativo e o servidor (linha 1). Na sequência, linhas 2 a 4, o servidor envia três códigos distintos, utilizando o canal TLS e dois canais *out-of-band*, neste caso SMS e e-mail, para o cliente. Os canais *out-of-band* são utilizados pois assume-se que o atacante não possui os recursos necessários para comprometer todos os canais de comunicação simultaneamente, aumentando assim a confiabilidade na autenticação.

Vale ressaltar que, para fins de implementação prática do protocolo, é recomendado a utilização de n canais *out-of-band* disponíveis nos dispositivos dos usuários finais. Além disso, é recomendado utilizar apenas os canais que não levem a problemas de usabilidade. Por exemplo, o código por e-mail pode reduzir a usabilidade da solução. Para resolver o problema, pode-se utilizar apenas SMS ou também algum outro canal complementar

(*e.g.*, mensagem automática via WhatsApp[10]).

A primeira chave temporária K_{t1} é gerada a partir da chave de sessão TLS e dos três códigos enviados pelo servidor. É utilizada uma função de *hash* criptográfica para gerar a chave (linha 5). A chave K_{t1} é então utilizada para cifrar (representado por E) o número de Identificação Internacional de Equipamento Móvel (*International Mobile Equipment Identity (IMEI)*) e o número pseudo-aleatório *app_rand1*. A mesma chave é utilizada para assinar a mensagem (*i.e.*, computar o HMAC) que é enviada do cliente para o servidor. Os valores do IMEI, do *app_rand1* e da chave K_{t1} são utilizados para gerar a segunda chave temporária K_{t2}. Assume-se que essa segunda chave é mais forte já que ela inclui um número global único (o IMEI do dispositivo) e um número pseudo-aleatório do aplicativo (o *app_rand1*). Isto, naturalmente, aumenta a entropia da chave K_{t2}.

Finalmente, o servidor envia um número pseudo-aleatório *server_rand* para o cliente (linha 8) e ambos geram a chave mestra K_m de alta entropia (linha 9). Entretanto, ainda resta verificar a chave K_m para finalizar o algoritmo. O usuário envia um número pseudo-aleatório *app_rand2*, cifrado, para o servidor (linha 10). O servidor decifra o número pseudo-aleatório recebido utilizando a sua chave K_m, incrementa em um (+1), cifra o novo valor e envia para o cliente (linha 11). Se o cliente conseguir validar o valor recebido, significa que as chaves são iguais e a execução do protocolo é finalizada com sucesso.

2.3 Protocolo de verificação

Um protocolo diferente para a geração de códigos únicos, que pode ser utilizado tanto para autenticação e/ou autorização, também faz parte da solução proposta. Como o protocolo de registro do aplicativo gera uma chave mestra K_m de alta entropia, conforme exemplificado anteriormente, a chave do gerador de códigos únicos pode ser derivada da chave mestra. Para a derivação, podem ser utilizadas funções de *hash* criptográfica seguras, como as dos grupos SHA2 e SHA3. A chave inicial de geração dos códigos únicos de autenticação pode ser tão simples quanto $K_c = \mathrm{H}(K_m||K_c)$. Como a chave K_c inicia vazia, a primeira K_c é igual a $\mathrm{H}(K_m)$.

Os OTACs são derivados da chave K_c. Para sincronizar a geração desses códigos, é necessário utilizar os índices iA, no aplicativo, e o iS, no servidor. No início da geração, OTAC $= K_c$ e K_c é evoluída para o próximo valor, i.e., $K_c = \mathrm{H}(K_m||K_c)$. Já os códigos OTACs são gerados da seguinte forma: OTAC $= \mathrm{H}^N(\mathrm{OTAC})$, no qual a função de *hash* criptográfica pode ser aplicada N vezes para gerar N códigos de autenticação únicos, distintos e com a propriedade de segurança denominada *Perfect Forward Secrecy* (PFS). Isso significa que não é possível descobrir os códigos OTACs passados a partir do código OTAC atual. Essa propriedade pode ser assegurada através da propriedade de

[10] <https://www.whatsapp.com/>

irreversibilidade das funções de *hash* criptográfica.

Para exemplificar a utilização dos códigos OTACs, supondo que os índices iA e iS estão com os valores 1 e 0, respectivamente, basta o aplicativo enviar uma mensagem com o índice atual do OTAC local para realizar a autenticação no servidor. Por exemplo, o aplicativo poderia enviar a mensagem `[GET, nome_do_arquivo,` *nonce*`, iA],HMAC` para o servidor. O HMAC é gerado utilizando como chave o OTAC do aplicativo. O servidor irá atualizar seu OTAC para OTAC = H^{iA-iS}(OTAC), utilizando o valor do índice recebido do aplicativo. Com o novo valor do OTAC, o servidor irá verificar a assinatura HMAC da mensagem. Se a assinatura conferir, o servidor confirmará a autenticação do aplicativo.

2.4 Exemplos de casos de uso

Por ser uma solução de propósito geral, a S3AS pode ser utilizada em diferentes cenários. A seguir são apresentados quatro casos de uso.

Controle de acesso em catracas eletrônicas

Catracas eletrônicas para controle de acesso estão cada vez mais comuns. Academias, restaurantes universitários, entre outros estabelecimentos, utilizam esses dispositivos para controle de acesso. Entretanto, uma boa parte desses estabelecimentos adota o modelo de utilização de dados estáticos (*e.g.*, CPF em um QR Code ou código de barras estático) para o controle de acesso.

Assumindo catracas simples e capazes de ler QR Code ou código de barras, OTACs podem ser utilizados para o controle de acesso. Considerando que cada indivíduo do sistema possui uma carteirinha digital de identificação, o usuário abre o aplicativo que gera automaticamente um novo QR Code contendo um novo e único OTAC de autenticação, como pode ser observado nas linhas 1 e 2 da Tabela 2.

Tabela 2: OTACs em catracas eletrônicas.

1. Usuário	Abre o aplicativo de identificação
2.	QR Code = [id, iA], HMAC_{OTAC}
3.	Aproxima o QR Code do leitor da catraca
4. Catraca	Lê o QR Code
5.	Atualiza o OTAC $\leftarrow \mathrm{H}^{iA-iS}$(OTAC)
6.	Verifica HMAC utilizando o OTAC como chave

Fonte: Elaborado pelo autor.

O usuário aproxima o QR Code do leitor (linha 3) e a catraca realiza a leitura (linha 4). A catraca atualiza então o OTAC (linha 5) e verifica o HMAC utilizando

o OTAC atualizado (linha 6), liberando ou não o acesso do usuário. Como pode ser observado, o processo é simples, eficiente e pode ser realizado off-line uma vez que não há nenhuma dependência de comunicação (acesso remoto via rede), de hora ou sincronização de relógios entre os dispositivos para a geração, leitura e validação dos QR Code.

Gerenciamento de chaves

O gerenciamento de chaves de acesso no campus Alegrete da UNIPAMPA é um problema interessante. Atualmente, a portaria é responsável por gerenciar as chaves, encarregando-se da entrega e recebimento de chaves dos docentes, Técnicos-administrativos em educação (sTAEs) e alunos autorizados. O controle é feito de forma manual e as regras são as seguintes: (a) docentes e TAEs podem retirar qualquer chave, com algumas excessões (*e.g.*, laboratórios específicos); e (b) alunos autorizados, identificação em listas ou documentos específicos, podem retirar chaves específicas, i.e., de determinadas salas ou laboratórios.

Como funciona a identificação? Docentes e TAEs simplesmente chegam até o porteiro, dizem o nome e a função (docente ou TAE) e retiram a chave desejada. Não há qualquer tipo de controle de identificação ou autenticação. No caso dos alunos, o nome do aluno é conferido em listas e documentos de autorização. Novos porteiros, ou porteiros temporários (*e.g.*, guardas nos finais de semana), não tem como conhecer todos os docentes e funcionários. Mesmo porteiros antigos tem dificuldade em conhecer todos os docentes e TAEs.

Como o processo de gerenciamento de chaves pode ser automatizado? Considerando um sistema simples de identificação baseado em carteirinhas digitais, a utilização do OTAC é detalhada no algoritmo da Tabela 3. No aplicativo da carteirinha digital do usuário, similar ao que ocorre na CNH (Carteira Nacional de Habilitação) digital[11], basta ter um QR Code contendo dados como: tipo da operação (*e.g.*, AUTH), nome do usuário, tipo de vínculo, foto em baixa resolução e o índice OTAC, como pode ser visto na linha 2 da Tabela 3. O QR Code é gerado e atualizado automaticamente pelo aplicativo.

Tabela 3: OTACs no gerenciamento de chaves.

1. Usuário	Abre o aplicativo da carteirinha digital de identificação
2.	QR Code = [AUTH, id, nome, tipo, foto, iA], HMAC_{OTAC}
3. Porteiro	Lê o QR Code
4.	Atualiza o OTAC $\leftarrow \text{H}^{iA-iS}$(OTAC)
5.	Verifica HMAC utilizando o OTAC como chave
6.	Confere informaçõe do usuário (nome, tipo, foto, etc.)

Fonte: Elaborado pelo autor.

[11] <https://www.detran.rs.gov.br/habilitacao-cnh/servicos/1037>

O porteiro, utilizando um aplicativo de validação, lê o QR Code do usuário (linha 3). O aplicativo atualiza o OTAC local (linha 4) e verifica o HMAC utilizando o OTAC atualizado como chave (linha 5). Se o HMAC coincidir, os dados do usuário são apresentados ao porteiro, validando a autenticação do usuário (linha 6). Como pode ser observado, neste caso de uso não há comunicação entre o cliente (aplicativo de carteirinha digital do usuário) e o servidor (aplicativo de validação do porteiro). É suficiente que o aplicativo do porteiro leia o QR Code do usuário para validar os dados utilizando OTACs. Em resumo, tudo isto pode ser realizado *off-line*. Basta o aplicativo do porteiro possuir uma *cache* de OTACs iniciais de autenticação (lado do servidor) dos usuários da instituição.

Aplicações em modelo cliente-servidor

O conceito de OTAC pode ser utilizado, teoricamente, em qualquer sistema que segue o modelo cliente-servidor. Como pode ser observado no algoritmo da Tabela 4, o cliente envia uma requisição ao servidor (linha 1). Esta requisição contém o índice iA do cliente e uma assinatura HMAC que utiliza o OTACs do cliente como chave. O servidor atualiza o seu OTAC local de acordo com o índice recebido do cliente (linha 2) e verifica o HMAC utilizando o novo OTAC como chave (linha 3). Se o HMAC coincidir, a requisição é considerada como válida.

Na sequência, o servidor atualiza o seu OTAC local e envia a imagem e o seu índice iS para o cliente (linha 4). Ao receber a mensagem, o cliente atualiza o seu OTAC local de acordo com o índice recebido do servidor (linha 5) e verifica o HMAC utilizando o novo OTAC como chave (linha 6). Se o HMAC coincidir, a mensagem é efetivamente recebida e processada. O servidor ainda envia uma mensagem de notificação (NOTIFY) de atualização da imagem. O cliente apenas repete o processo de atualização de OTAC local e verificação do HMAC.

Este exemplo simples ilustra o funcionamento e aplicação do conceito de OTAC a aplicações modelo cliente-servidor. No exemplo apresentado, é utilizado um código de autenticação por mensagem, o que é considerado o caso mais robusto (e desejável) em termos de segurança (Kreutz et al., 2018).

Tabela 4: OTACs em modelo cliente-servidor.

1. Cliente -> Servidor	[GET, nome_arq, *nonce*, iA], HMAC_{OTAC}
2. Servidor	OTAC $\leftarrow \text{H}^{iA-iS}$(OTAC)
3.	Verifica HMAC utilizando o OTAC como chave
4. Servidor -> Cliente	[REPLY, nome_arq, *nonce*, iS], HMAC_{OTAC}
5. Cliente	OTAC $\leftarrow \text{H}^{iS-iA}$(OTAC)
6.	Verifica HMAC utilizando o OTAC como chave
7. Servidor -> Cliente	[NOTIFY, *nonce*, iS], HMAC_{OTAC}
8. Cliente	OTAC $\leftarrow \text{H}^{iS-iA}$(OTAC)
9.	Verifica HMAC utilizando o OTAC como chave

Fonte: Elaborado pelo autor.

Automação e segurança residencial

A automação residencial é um mercado em ascensão em todo o mundo. No Brasil, cerca de 300 mil residências já possuem algum tipo de equipamento de automação residencial (AURESIDE, 2019). Empresas como a Amazon e a Google estão provendo soluções como o Amazon Echo e o Google Home para automatizar tarefas e serviços como a manipulação de lâmpadas, o controle de tomadas e a interação entre dispositivos (SHOWMETECH, 2019).

Com relação ao controle de acesso, fechaduras de nova geração já suportam tecnologias como NFC e Radio-Frequency Ientification (RFID)[12] (*Radio-Frequency IDentification*) para facilitar a automação. A Samsung oferece uma solução para fechaduras automatizadas que custa aproximadamente R$ 900 (Tecmundo, 2015a). A solução funciona com chaveiros com tags RFID que, ao aproximados da fechadura, liberam a porta.

Enquanto que *tags* RFID representam um problema para a implementação de OTACs, uma vez que uma tag permite apenas a definição de códigos estáticos, pré-definidos, tecnologias como NFC, que estão sendo ampla e rapidamente suportadas em *smart cards* e *smartphones*, podem ser utilizadas em fechaduras automatizadas. A tecnologia NFC viabiliza a utilização de OTACs no processo de autenticação e autorização de acesso em residências.

Com NFC e OTACs, como pode ser observado na Tabela 5, o usuário inicia uma comunicação com a fechadura enviando uma requisição de autenticação do tipo SAUTH com seu identificador e índice (linha 1). Em seguida, a fechadura atualiza o OTAC e verifica o HMAC (linhas 2 e 3). A fechadura libera o acesso e responde a solicitação do usuário através de uma mensagem do tipo ACK com seu identificador e índice (linhas 4 e

[12] <https://www.epc-rfid.info/rfid>

5). Em caso de acesso não autorizado, a mensagem seria do tipo NAK. Por fim, o usuário atualiza o OTAC e realiza a verificação (linhas 6 e 7).

Este processo é simples e rápido, o que é essencial para fechaduras residenciais. Não há necessidade de conexão com a Internet e não há dependência de sincronização de relógios, ou seja, pode ser totalmente *off-line*.

Tabela 5: OTACs em fechaduras.

1. Usuário -> Fechadura	[SAUTH, *nonce*, idU, iA], HMAC_{OTAC}
2. Fechadura	Atualiza o OTAC $\leftarrow \text{H}^{iA-iS}$(OTAC)
3.	Verifica HMAC utilizando o OTAC como chave
4.	Libera acesso
5. Fechadura -> Usuário	[ACK, *nonce*, idF, iS], HMAC_{OTAC}
6. Usuário	Atualiza o OTAC $\leftarrow \text{H}^{iS-iA}$(OTAC)
7.	Verifica HMAC utilizando o OTAC como chave

Fonte: Elaborado pelo autor.

Além de ser uma solução de propósito geral, pois os algoritmos apresentados mostram a aplicação em cenários com catracas ou até mesmo no gerenciamento de chaves, a S3AS mostra-se, também, uma solução que aplicável com diferentes tecnologias. Tomando como exemplo o **controle de acesso em catracas eletrônicas** (Seção 2.4) e o exemplo de **automação e segurança residencial** apresentado nesta Seção, as tecnologias utilizadas foram QR Code e NFC, respectivamente.

3 VERIFICAÇÃO AUTOMÁTICA DOS PROTOCOLOS

A verificação automática de propriedades de segurança é crucial para demonstrar a corretude dos protocolos. Ferramentas como a Scyther (CREMERS, 2006) contribuem nesse processo, como demonstrado recentemente pelos autores de forma didática e prática (JENUARIO et al., 2019a; JENUARIO et al., 2019b).

3.1 Verificação do protocolo de identificação

O Algoritmo 1 descreve o protocolo de identificação (Protocolo 1) na semântica da ferramenta Scyther. Os tipos predefinidos *Code*, *TemporaryKey* e *MasterKey* são declarados na linha 1. As chaves *tlssessionkey*, K_{t1}, K_{t2} e K_m são declaradas globalmente nas linhas 2, 5 e 6. Nas linhas 3 e 4 são definidas uma função de hash criptográfica H e uma função HMAC, respectivamente.

A chamada à função **protocol** marca o início da especificação do protocolo `S3AS` (linha 7), que contém quatro agentes, sendo eles (`KGC`, `MKG`, `Cliente` e `Servidor`) que possuem papéis (**role**) explícitos.

Como pode ser observado na semântica do algoritmo, cada evento de envio (*e.g.*, **send_1**) possui um evento de recebimento (*e.g.*, **recv_1**) correspondente (*e.g.*, linhas 9 e 14). A sintaxe do evento **send_1** indica que a transmissão é de `KGC` (*Key Generation Center*) para `MKG` (*Master Key Generation*), simulando a geração da chave de sessão *tlssessionkey*.

No agente `MKG` (linha 11), há um evento com a sintaxe idêntica ao `KGC`, cuja única diferença é o tipo, *i.e.*, **recv** ao invés de **send**, inicializando a chave de sessão *tlssessionkey* (linha 15). A primeira chave temporária K_{t1} é gerada através da função hash criptográfica H, que recebe como parâmetro a chave de sessão *tlssessionkey* e os códigos *code*1, *code*2 e *code*3. A chave K_{t1} dos agentes `Cliente` e `Servidor` é gerada nas linhas 16 e 17, respectivamente, garantindo que esta seja igual em ambos os agentes.

As variáveis *nonce*, *IMEI*, *appRand*1 e *appRand*2 (linhas 24 e 25), dos tipos **Nonce** e **tipoIMEI** e antecedidas por **fresh**, são criadas no agente `Cliente` e inicializadas automaticamente com valores pseudo-aleatórios (nas linhas 28 e 32). Já as variáveis *code*1, *code*2, *code*3 e *serverRand* (linhas 26 e 27), são **var** ao invés de **fresh**, o que significa que a variável será utilizada para armazenar valores recebidos (nas linhas 28 e 31).

Para determinar que um termo é secreto e autêntico (**Secret** e **Nisynch**), é necessário que hajam eventos de afirmação (**claim**), como pode ser observado nas linhas 35 à 38 e 52 à 55. Essas afirmações definem os requisitos de segurança do protocolo. No caso, as afirmações criadas verificam se as *chaves* geradas (K_{t1}, K_{t2} e K_m) permanecem secretas e autênticas durante as comunicações.

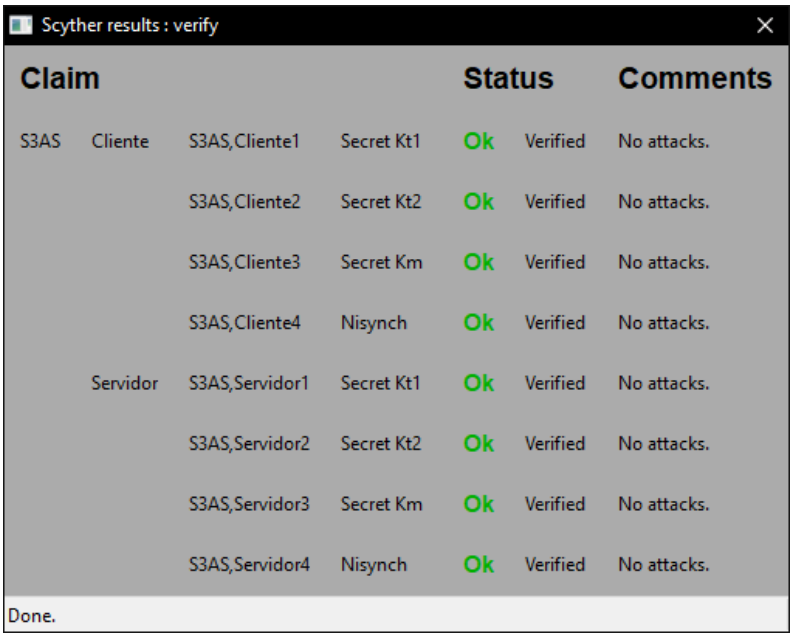
Ao verificar o protocolo com a ferramenta Scyther, é gerado um relatório que aponta se foram encontradas falhas. Quando há falhas, a ferramenta apresenta também um fluxograma que ilustra em detalhes como o ataque pode ser realizado ao protocolo.

Algoritmo 1: Protocolo de identificação na semântica da Scyther

```
1  usertype Code, TemporaryKey, MasterKey,
     tipoIMEI, Random;
2  secret tlssessionkey: SessionKey;
3  hashfunction H;
4  const HMAC: Function;
5  secret Kt1, Kt2: TemporaryKey;
6  secret Km: MasterKey;
7  protocol S3AS(KGC, MKG, Cliente, Servidor){
8    role KGC{
9      send_1(KGC,KDF,H(tlssessionkey));
10   }
11   role MKG{
12     fresh code1, code2, code3: Code;
13     fresh appRand1, serverRand: Random;
14     fresh IMEI: tipoIMEI;
15     recv_1(KGC,KDF,H(tlssessionkey));
16
     send_2(MKG,Cliente,Kt1(H(tlssessionkey,code1,
     code2,code3)));
17
     send_3(MKG,Servidor,Kt1(H(tlssessionkey,code1,
     code2,code3)));
18
     send_5(MKG,Cliente,Kt2(H(IMEI,appRand1,Kt1)));
19     send_6(MKG,Servidor,
     Kt2(H(IMEI,appRand1,Kt1)));
20     send_8(MKG,Cliente,Km(H(Kt1,Kt2,
     IMEI,appRand1,serverRand)));
21     send_9(MKG,Servidor,
     Km(H(Kt1,Kt2,IMEI,appRand1,serverRand)));
22   }
23   role Cliente{
24     fresh nonce, appRand1, appRand2: Nonce;
25     fresh IMEI: tipoIMEI;
26     var code1, code2, code3: Code;
27     var serverRand: Random;
28
     recv_2(MKG,Cliente,Kt1(H(tlssessionkey,code1,
          code2, code3)));
29     send_4(Cliente,Servidor,
     HMAC(nonce,{IMEI,appRand1}
     Kt1(Cliente,Servidor)));
30     recv_5(MKG,Cliente,Kt2(H(IMEI,
     appRand1,Kt1)));
31     recv_7(Servidor, Cliente,
     HMAC(nonce,{serverRand}Kt2(Servidor, Cliente)));
32
     recv_8(MKG,Cliente,Km(H(Kt1,Kt2,IMEI,
     appRand1,serverRand)));
33     send_10(Cliente,Servidor,
     HMAC(nonce,{appRand2}Km(Cliente,Servidor)));
34     recv_11(Servidor,Cliente,
     HMAC(nonce,{appRand2}Km(Servidor, Cliente)));
35     claim(Cliente,Secret,Kt1);
36     claim(Cliente,Secret,Kt2);
37     claim(Cliente,Secret,Km);
38     claim(Cliente,Nisynch);
39   }
40   role Servidor{
41     var nonce, appRand1, appRand2: Random;
42     var IMEI: tipoIMEI;
43     var code1, code2, code3: Code;
44     fresh serverRand: Random;
45
     recv_3(MKG,Servidor,Kt1(H(tlssessionkey,
     code1,code2, code3)));
46
     recv_4(Cliente,Servidor,HMAC(nonce,{IMEI,
     appRand1}Kt1(Cliente,Servidor)));
47
     recv_6(MKG,Servidor,Kt2(H(IMEI,appRand1,
     Kt1)));
48     send_7(Servidor, Cliente,
     HMAC(nonce,{serverRand}Kt2(Servidor, Cliente)));
49     recv_9(MKG,Servidor,
     Km(H(Kt1,Kt2,IMEI,appRand1,serverRand)));
50     recv_10(Cliente,Servidor,
     HMAC(nonce,{appRand2}Km(Cliente,Servidor)));
51     send_11(Servidor,Cliente,
     HMAC(nonce,{appRand2}Km(Servidor, Cliente)));
52     claim(Servidor,Secret,Kt1);
53     claim(Servidor,Secret,Kt2);
54     claim(Servidor,Secret,Km);
55     claim(Servidor,Nisynch);
56   }
57 }
```

Para o caso do código do algoritmo de identificação, como pode ser observado na Figura 2, a comunicação entre os agentes `Cliente` e `Servidor` é segura segundo a análise automática da ferramenta. Não há nenhum indicativo de falha no *Status* do relatório, isto é, tudo está como *Ok*, verificado e confirmado como sem ataques.

Figura 2: Verificação do Protocolo de Identificação (Algoritmo 1).



Fonte: Elaborado pelo autor.

3.2 Verificação do protocolo de autenticação

O Algoritmo 2 descreve o protocolo de autenticação (Protocolo 2) na semântica da Scyther. Nas linhas 1 a 4 é definido o tipo **UniqueCode** para representar o OTAC, uma função hash criptográfica H e uma função HMAC, respectivamente. O processo de autenticação é iniciado com a atualização do OTAC do usuário, que será utilizado na autenticação com a catraca.

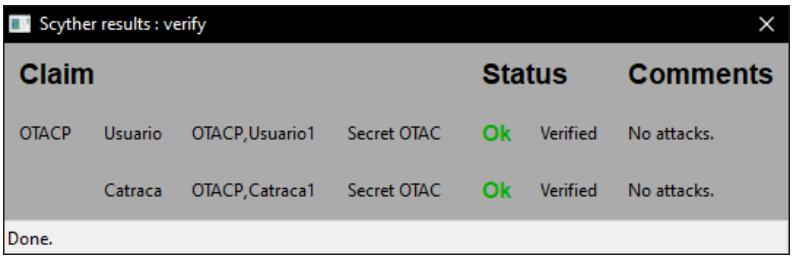
O agente `KGC` gera e envia um novo código ao usuário (linha 7). Na sequência, o usuário envia seu `id`, `iA` e o `HMAC` da mensagem (linha 14) para a catraca. Vale ressaltar que, por limitações da ferramenta, não é possível representar a diferença de índices (*e.g.*, $iA-iS$) do algoritmo apresentado na Seção 2.4. Para superar essa limitação, foi implementada uma abstração onde a catraca recebe e atualiza o seu código OTAC através da KGC (linhas 19 e 20). Por fim, são declarados os eventos de afirmação (linhas 15 e 21) para determinar se os OTACs dos dois agentes permanecem seguros durante a análise automática de segurança da Scyther.

A Figura 3 apresenta o resultado da análise da Scyther. Como pode ser observado, a análise retornou *Ok* (no *Status*) para as afirmações do usuário e da catraca, indicando que os OTACs estão seguros.

Algoritmo 2: Protocolo de autenticação na semântica da Scyther

```
1  usertype UniqueCode;
2  hashfunction H;
3  secret OTAC: UniqueCode;
4  const HMAC: Function;
5  protocol OTACG(KGC,Usuario,Catraca){
6     role KGC{
7        send_1(KGC,Usuario,H(OTAC));
8        send_3(KGC,Catraca,H(OTAC));
9     }
10    role Usuario{
11       fresh id, iA: Nonce;
12       var nr: Nonce;
13       recv_1(KGC,Usuario,H(OTAC));
14       send_2(Usuario,Catraca,id,iA,HMAC(id,iA));
15       claim(Usuario,Secret,OTAC);
16    }
17    role Catraca{
18       var iA, id: Nonce;
19       recv_2(Usuario,Catraca,id,iA,HMAC(id,iA));
20       recv_3(KGC,Catraca,H(OTAC));
21       claim(Catraca,Secret,OTAC);
22    }
23 }
```

Figura 3: Verificação do protocolo de autenticação.



Fonte: Elaborado pelo autor.

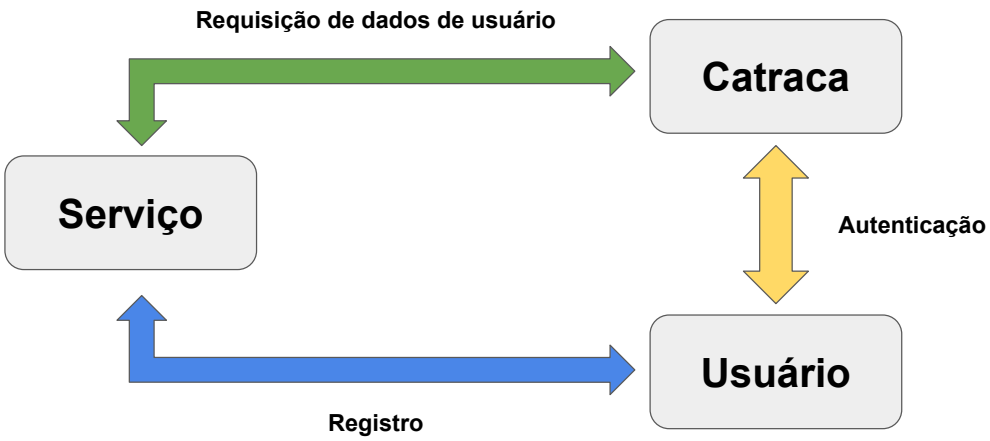
4 AVALIAÇÃO EXPERIMENTAL DOS PROTOCOLOS

Para realizar medições de desempenho, foi implementado um protótipo simulando autenticações através de catracas eletrônicas, como ilustrado na Figura 4. O protótipo é baseado em três agentes, sendo eles o serviço, o usuário e a catraca. O serviço é responsável por gerenciar o registro de usuários e o controle de acesso das catracas. Os dados registrados são transferidos para a catraca e, em caso de erros na autenticação, esta notifica o serviço.

O usuário se registra através do serviço e solicita autenticação à catraca através de um QR Code composto por um identificador e um índice, como no exemplo apresentado na Seção 2.4. Quando o usuário ainda não é reconhecido, a catraca solicita os dados atualizados do usuário ao serviço.

A implementação utiliza a função de *hash* criptográfica SHA-256 para calcular um OTAC, ou seja, cada código possui 256 bits de tamanho, e criptografia AES para cifrar os dados. É simulada uma comunicação bloqueante entre a catraca e o usuário no qual o usuário espera a reposta da catraca. A troca de informações entre catraca e usuário é realizada através da utilização de QR Codes.

Figura 4: Arquitetura do protótipo de simulação de autenticação.



Fonte: Elaborado pelo autor.

Os testes foram realizados em uma máquina que possui um processador i7-4510U dual-core de 2,0 GHz, 8Gb de memória RAM, 1Tb de disco rígido, placa de vídeo integrada Intel 4400 executando o sistema operacional Linux Debian 10 Buster.

As simulações foram realizadas considerando três casos de avaliação:

Latência de registro de um usuário, envolvendo o tempo necessário para a geração da chave mestra conforme especificado no Protocolo 1. Esta simulação mede o tempo de comunicação entre o usuário e o servidor, o tempo de geração de chaves temporária e de geração de chave mestra.

Latência de autenticação sequencial *on-line* na catraca representa o tempo necessário para realizar a geração e verificação de OTACs de forma sequencial, a latência de comunicação e a leitura e manipulação de QR Code.

Latência de autenticação sequencial *on-line* no usuário representa o tempo de geração e verificação de OTACs de forma sequencial, a latência de comunicação e a geração, leitura e manipulação de QR Code.

Latência de autenticação sequencial *off-line* na catraca representa o tempo necessário para realizar as operações de geração e verificação de OTACs de forma sequencial e a leitura e manipulação de QR Code.

Latência de autenticação sequencial *off-line* na catraca e no usuário representa tempo de geração e verificação de OTACs de forma sequencial e a geração, leitura e manipulação de QR Code.

Para calcular a **latência de registro**, foram registrados 10 mil usuários, utilizando um canal *out-of-band* que simula o envio de um código por SMS. O canal foi implementado através de um *socket Transmission Control Protocol/Internet Protocol (TCP/IP)* que é utilizado apenas para enviar o código.

Nas avaliações sequenciais foram realizadas 10 mil autenticações com o índice do OTAC limitado a 20000. A média é calculada pela equação 4.1, no qual TOT é o tempo necessário para calcular um OTAC e TOO é o tempo de latência das demais operações. NA é o número de autenticações.

$$\frac{\sum_{i=1}^{n}(TOT_i + TOO_i)}{NA} \tag{4.1}$$

A equação 4.2 representa as operações que compõe TOO, sendo elas: geração (TGQ), leitura e manipulação de QR Code (TLM), latência de comunicação (TCO) e verificação do OTAC (TVO), respectivamente. O menor tempo é gasto no cálculo na verificação dos OTACs com média de 0,03ms, já o maior tempo é gasto na latência de comunicação. Com isto, espera-se que as autenticações *on-line* levem maior tempo.

TOT é o tempo necessário para calcular um OTAC (0,03ms) e TOO é o tempo de latência das demais operações, como geração (9,17ms), leitura e manipulação de QR Code (5.34ms), comunicação (49,59ms) e verificação do OTAC (0,04ms). Por fim, NA é o número de autenticações.

$$TOO = TGQ + TLM + TCOTVO \tag{4.2}$$

A Tabela 6 apresenta os valores de média (M), variância (V) e desvio padrão (D) para a **latência de registro** e para as **avaliações sequenciais**. Para o cômputo da média, foram eliminados 5% dos melhores e dos piores casos da amostragem total de 10 mil autenticações. Como pode ser observado na tabela, o processo mais lento, mas que ocorre uma única vez, é o de registro, cuja média é de 88,33ms. Este tempo é baixo considerando que 100ms é um número aceitável para sistemas de autenticação (KREUTZ et al., 2016).

Nas avaliações de autenticação, as avaliações *on-line* são as que possuem o maior tempo. Isto é justificado por considerar a latência de comunicação entre usuário e catraca, que é em media 49,59ms. Vale ressaltar que, as avaliações *on-line* possuem valores similares porque a implementação utiliza comunicação bloqueante, ou seja, o usuário fica esperando a resposta (*i.e.*, solicitação de autenticação autorizada ou rejeitada) da catraca.

Já para as avaliações de autenticação *off-line*, há variação nos valores. Isto se deve ao fato que os valores da **autenticação sequencial *off-line* na catraca** são medidos considerando apenas as operações de geração (0,03ms) e verificação de OTAC (0,04ms) e a leitura e manipulação (5,34ms). Para a **autenticação sequencial *off-line* na catraca e no usuário** também é considerado o valor de geração de QR Code, que tem como média 9,17ms. A utilização de outra tecnologia, como NFC, ao invés de QR Code poderia reduzir o tempo para as avaliações.

Os tempos para as médias para as autenticações *off-line* são mais baixos A**autenticação *off-line* na catraca** obteve melhor desempenho com apenas 5,34ms. A média para este caso é baixa, pois não mede o tempo de geração de QR Code. Como o tempo médio para calcular um OTAC (TOT) é de apenas 0,03ms, pode-se concluir que o maior tempo é gasto nas demais operações (TOO). No caso da autenticação *off-line* parcial, as operações que mais consomem tempo são a de leitura e manipulação de QR Code (5,34ms) e a verificação do OTAC (0,04ms). Em outras palavras, TOO = 5,43 - 0,03 = 5,40ms. A solução torna-se interessante para cenários que a Internet seja dispensável ou que seja requisito não haver conexão. Cenários como o SESC-RS, academias ou automação residenciais são exemplos que a S3AS pode ser aplicada de forma *off-line* para oferecer usabilidade e autenticação segura.

Tabela 6: Avaliações sequenciais.

Cenário	**M**	**V**	**D**
Latência de registro	88,33	0,87	0,94
Latência de autenticação sequencial *on-line* na catraca	54,91	4,22	2,06
Latência de autenticação sequencial *on-line* no usuário	54,96	4,24	2,06
Latência de autenticação sequencial *off-line* na catraca	5,43	1,24	1,12
Latência de autenticação sequencial *off-line* na catraca e no usuário	12,55	3,59	1,90

Fonte: Elaborado pelo autor.

Para a avaliação de **latência de autenticação com múltiplas catracas**, foram simuladas 3 catracas simuladas configuradas com 2 ciclos de no máximo 100 índices cada, ou seja, cada ciclo comporta no máximo 100 autenticações. Se o índice do usuário se distanciar mais de 200 (dois ciclos completos) do índice mantido pela catraca, a autenticação será negada. A escolha das catracas é feita pseudo-aleatoriamente e a quantidade de autenticações sequenciais que o usuário faz também, em um intervalo de 10 a 300 autenticações. A avaliação foi realizada dessa forma para simular um cenário real em que não se sabe em qual catraca será realizada a autenticação ou quantas autenticações um usuário irá realizar em cada uma delas.

A Tabela 7 apresenta os valores de média, variância e desvio padrão, em milissegundos, para o cenário da **latência de autenticação com múltiplas catracas**. As medições foram realizadas considerando os critérios utilizado na **autenticação sequencial *off-line* parcial**. Como pode ser observado, o tempo médio de autenticação e o desvio padrão permaneceram similares aos apresentados na Tabela 6, para autenticação sequencial *off-line*. Adicionalmente, são apresentados os números de autenticações, falhas e número de ciclos completos por catraca.

As catracas registraram 13, 17 e 27 falhas, respectivamente. Casualmente o número de autenticações seguidas que um usuário realiza na mesma catraca pode ser a causar disto. Quanto mais autenticações realizadas de forma seguida por um usuário, menos falhas. Quando menos autenticações, mais falhas. O que acontece é que o usuário autenticou mais vezes nas catracas 1 e 2. Com isso, o índice do OTAC passou de 200, o limite pré-definido, mais vezes na catraca 3, que recebeu menos requisições de autenticação do usuário.

Tabela 7: Avaliação com múltiplas catracas.

	Catracas		
	Catraca 1	**Catraca 2**	**Catraca 3**
Média	5,60	5,53	5,57
Variância	1,50	1,47	1,48
Desvio Padrão	1,23	1,22	1,22
# Autenticações	3839	3715	2446
# Falhas	13	17	27
# Ciclos	97	99	98

Fonte: Elaborado pelo autor.

5 DISCUSSÕES

Vínculo único. O usuário pode utilizar apenas um único dispositivo por vez, o que reduz os riscos de segurança. Exemplo: um atacante, mesmo possuindo todos os dados do usuário (e.g., credenciais de acesso), não irá conseguir registrar e utilizar o aplicativo e os dados do usuário, em outro smartphone, enquanto o registro do aplicativo do usuário estiver ativo. Para migrar de dispositivo, o usuário é obrigado a primeiro revogar o registro do aplicativo no dispositivo atual. Este tipo de procedimento de segurança é adotado por diferentes bancos internacionais e nacionais, como o Revolut, N26, Banco e NuBank. No caso da S3AS, o protocolo de identificação utiliza as credenciais do usuário e o IMEI para geração da chave mestra e consequente vínculo ao dispositivo.

Robustez da chave mestra. O protocolo foi projetado para que a chave mestra possua alta entropia. A ideia é evitar que um atacante consiga comprometer o canal de comunicação primário. A quantidade de canais para transmissão de dados no processo de vinculação pode ser configurada, de acordo com a necessidade do cenário em que será aplicada, sendo uma variável interessante para garantir maior robustez. Os códigos pseudo-aleatórios utilizados para gerar as chaves temporárias são enviados através de diferentes canais de comunicação.

Confiabilidade de códigos únicos. Códigos únicos, não re-utilizáveis, com alta entropia, como os OTACs, são mais confiáveis do que dados biométricos, que são estáticos por natureza. Dados biométricos são iguais a um código (ou uma senha) que você não pode trocar. Se o código estático imutável vazar, toda a segurança, em todos os sistemas baseados exclusivamente nela, será comprometida. Por ser estática, na prática, autenticação biométrica deveria ser evitada como único ou principal fator de autenticação.

Perfect Forward Secrecy (PFS) é uma propriedade de segurança que tem por objetivo garantir a confidencialidade de dados de comunicações passadas. No caso de protocolos que geram códigos de autenticação únicos e garantem PFS, é assegurada a irreversibilidade dos códigos gerados, ou seja, a partir do OTAC atual, um ataque não consegue descobrir os códigos passados. O gerador de OTAC, proposto no protocolo de verificação (ver Seção 2.3), utiliza funções *hash* criptográficas para garantir a propriedade de PFS. Funções *hash* criptográficas, por definição, não permitem descobrir os dados de entrada a partir dos dados de saída, isto é, não há como descobrir o OTAC anterior (que faz parte da entrada da função) a partir do OTAC atual, que é a saída da função.

Privacidade dos códigos de verificação. Ao garantir a propriedade de PFS através de funções *hash* criptográficas, garante-se também a privacidade do usuário com relação aos códigos de autenticação propriamente ditos. Mesmo tendo acesso aos OTACs em transito (e.g., sendo utilizado entre uma catraca e o dispositivo do usuário), um atacante não consegue deduzir nada sobre o usuário, como quantas vezes ele já se autenticou naquele ambiente ou quem é o usuário do código. Isto é garantido pelas funções *hash* criptográficas. Uma segunda característica importante destas funções é o fato de gerarem uma saída

pseudo-aleatória, ou seja, que não há como relacionar com os dados de entrada ou outra informação qualquer do usuário no protocolo proposto. Diferentemente de códigos de soluções como a do Cartão Virtual do SESC-RS (que é o QR Code do CPF do associado), um OTAC é simplesmente um código sem significado algum para um observador externo.
Utilização de diferentes tecnologias. Como apresentado na Seção 2.4, os protocolos propostos na S3AS são agnósticos de tecnologia de comunicação, isto é, podem ser utilizados em diferentes casos de uso e com o suporte de diferentes tecnologias. Por exemplo, na prática, os códigos de autenticação OTACs podem ser utilizados com dispositivos que possuem apenas leitores de QR Code ou sistemas que utilizam tecnologias de comunicação como NFC e Bluetooth.
Utilização de fatores biométricos. A autenticação biométrica diz respeito a características físicas de um usuário como impressão digital, reconhecimento de íris, reconhecimento de retina, reconhecimento de face e até mesmo impressão de voz. A autenticação biométrica funciona de forma similar a uma senha estática, pois é quase impossível alterar algumas das características citadas. Logo, vazamentos de dados biométricos podem ser um risco ainda maior.

Recentemente a empresa Suprema sofreu um vazamento de dados biométricos de quase 1 milhão de usuários europeus (Olhar Digital, 2019), incluindo informações de reconhecimento facial, nomes de usuário e senhas não criptografadas. O governo indiano também sofreu um vazamento em sua base de dados Aadhaar (Tecmundo, 2015b). A leitura de íris e impressões digitais de cerca de 130 milhões de pessoas está exposta e suscetível a fraudes. As informações biométricas destes usuários estarão vulneráveis já que não é possível fazer alterações faciais de forma trivial, ou seja, são um tipo de senha que não pode ser trocada.

A utilização de biometria como um único fator pode trazer um grande risco. Isto é evidenciado pela fraude em restaurantes universitários, que utilizavam biometria como único fator de autenticação, causando um prejuízo de 3 milhões de reais (G1, 2016).

A S3AS oferece uma opção segura de autenticação sem a utilização de dados biométricos. A utilização dos OTACs de alta entropia faz com que a autenticação seja mais robusta.
Aplicação real em catracas eletrônicas As catracas eletrônicas são utilizadas no-SESC-RS e em outros locais como academias e empresas para controle de acesso. Há modelos em que a velocidade de operação é de 30 pessoas por minutos, possuindo 256 Kbytes de memória e sendo integrável com sistemas escolares e com sistemas de controle de acesso de veículos (SLSU, 2018). Considerando que um OTAC ocupa 256 bits, seria possível armazenar códigos de 8000 neste modelo de catraca. Outros modelos podem suportar até 50 mil (Perco, 2017) identificadores de usuários. Levando em conta tamanho de um OTAC, seria necessário apenas 1,6 Mbytes de memória para armazenar os dados de todos eles. Estudos mostram que é possível utilizar a função MD5 modificada garantindo

propriedades de segurança (GUZMAN; SISON; MEDINA, 2018). Se aplicada, seriam gerados OTACs com 128 bits de tamanho, reduzindo pela metade o tamanho dos códigos. Há também implementações compactas de funções *hash* utilizando Field Programmable Gate Array (FPGA), reduzindo pela metade o seu tamanho e mantendo propriedades de segurança e performance em relação ao custo de energia (Lara-Nino; Morales-Sandoval; Diaz-Perez, 2018), ideal para utilização em *hardwares* como catracas eletrônicas.

Utilizar *hardwares* com mais memória para comportar códigos de 50000 mil usuários para a aplicação da S3AS pode ser interessante, pois há maior robustez em termos de segurança e tempo de operação mais baixo.

Furto ou comprometimento de dispositivos. O limite e a quantidade autenticações, respectivamente, são formas que a S3AS utiliza para identificar um ataque de saturação e possivelmente um dispositivo comprometido ou furto. Se um usuário repentinamente começa a realizar mais autenticações do que a sua média ou até mesmo atinge o limite de autenticações por vezes seguidas, pode ser um indício de comprometimento e aquele usuário é bloqueado. Vale ressaltar que um usuário com dispositivo comprometido ou furtado também pode acusar o fato para revogar as credenciais e as atividades da aplicação naquele dispositivo.

Há outras formas de detecção como análise do padrão comportamental do usuário na aplicação. Padrões de digitação, horário que o usuário costuma utilizar a aplicação e até mesmo o padrão de navegação dentro da aplicação podem ser formas de identificar um dispositivo comprometido. Para trabalhos futuros, a pesquisa sobre estes padrões pode ser aplicada a S3AS.

Utilização da solução *off-line*. A S3AS pode ser utilizada de forma *off-line*, como comentado no Capítulo 4. A autenticação *off-line* pode ser interessante em áreas isoladas, em que não há conexão com a Internet ou quando há necessidade de autenticação de forma mais rápida.

Uma vantagem da aplicação *off-line* é evitar a captura de códigos através da Internet. O aumento no desempenho é outra vantagem, pois autenticações *off-line* são mais rápidas por não haver latência de comunicação, como mostrado na Seção 4.

Em contrapartida, uma desvantagem é a falta de sincronização entre as catracas. Considerando um cenário com múltiplas catracas como apresentado na Seção 4, a falta de comunicação entre catracas faz com que uma catraca não tenha informação de que um OTAC já foi utilizado em outra catraca.

6 SOLUÇÕES DE IDENTIFICAÇÃO E VERIFICAÇÃO

A Tabela 8 resume diferentes soluções de identificação e verificação que utilizam dispositivos móveis, tokens, *smartcards*, entre outros recursos, como segundo ou terceiro fator de autenticação. A maioria dessas soluções é projetada especificamente para *Internet Banking* e sistemas bancários (*e.g.*, ATMs), priorizando segurança em detrimento de usabilidade. Exceto a solução proposta (S3AS), todas utilizam múltiplos fatores de autenticação (alguns mais simples, outros mais complexos) e são de domínio específico.

Três soluções apresentadas na Tabela 8 utilizam Public Key Encryption (PKE) (Pratama; Prima, 2016; Putra; Sadikin; Windarta, 2017; Starnberger; Froihofer; Goeschka, 2009). Este conceito é comumente conhecido como criptografia de chave pública ou criptografia assimétrica (GLOBALSIGN, 2019). Em sistemas PKE, há duas chaves, uma privada e outra pública. A chave pública pode ser compartilhada publicamente e utilizada para cifrar mensagens que somente o destinatário, com respectiva chave privada, conseguirá decifrar.

As soluções baseadas em PKE utilizam o modelo desafio-resposta. Supondo o uso de dois pares de chaves, um número pseudo-aleatório é cifrado com a chave pública do usuário, assinado com a chave privada do servidor e depois enviado como um desafio ao usuário. Em seguida, para enviar a resposta ao servidor, o usuário deverá utilizar a sua chave privada e a chave pública do servidor para decifrar o valor. Se a resposta estiver correta, o usuário é autenticado.

Outras soluções utilizam dois índices inteiros compartilhados entre o usuário e a aplicação para gerar códigos únicos através de duas funções *hash* compostas e o modelo desafio-resposta Eldefrawy, Alghathbar e Khan (2011). Este tipo de solução é implementada em casos onde o usuário se autentica por um computador através do primeiro fator de autenticação (usuário/senha). No login é enviada uma requisição de autenticação ao servidor. O servidor responde à requisição com dois valores que o usuário deve utilizar num aplicativo específico do seu *smartphone* para calcular a resposta. O código gerado pelo aplicativo deverá ser digitado na aplicação de seu computador para completar a autenticação. Esse processo é similar a autenticação de dois fatores utilizada por sistemas com o Gmail[1] e o Facebook[2].

A solução proposta neste trabalho é de propósito geral e pode ser utilizada para identificação e verificação (*i.e.*, autenticação, autorização) em aplicativos de *smartphones* como o Cartão Virtual do SESC-RS. A verificação é realizada sem a necessidade de canais *out-of-band*. Em outras palavras, S3AS é, até onde se sabe, a única solução que assegura uma verificação mais robusta utilizando apenas um único fator de autenticação. Assumindo que o acesso ao *smartphone* é suficientemente seguro, o uso da identidade do usuário estará protegida mesmo em casos de perda ou roubo do dispositivo móvel.

[1] <http://gmail.com/>
[2] <https://www.facebook.com/>

Tabela 8: Soluções de identificação e verificação robustas.

Solução	**Projetada para**	**Autenticação**	***Canal(is) out-of-band***
(Eldefrawy; Alghathbar; Khan, 2011)	*Internet Banking*	Usuário/senha e OTP via aplicação móvel	OTP é enviado para dispositivo móvel
(Pratama; Prima, 2016)	*Internet Banking*	Usuário/senha e OTP via aplicação móvel	*QR Code* no terminal eletrônico ou PC
(KHAMIS et al., 2017)	Displays públicos e ATMs	Senha (PIN) e dispositivo móvel	*Bluetooth*
(Putra; Sadikin; Windarta, 2017)	*Internet Banking*	Usuário/senha, OTP, PIN e *smart card*	NFC
S3AS	Aplicativos de identificação	Autenticação de fator único utilizando OTACs	Códigos enviados através de SMS e e-mail

Fonte: Elaborado pelo autor.

7 CONSIDERAÇÕES FINAIS

S3AS é uma solução de autenticação e autorização composta por dois protocolos principais, um de vinculação das credenciais do usuário ao dispositivo e outro de geração de códigos únicos. Os protocolos propostos podem ser utilizados para incrementar a segurança de sistemas, sem penalizar a usabilidade, utilizando apenas um único fator dinâmico e robusto de autenticação e autorização.

Os protocolos seguem os princípios dos OTACs, que promovem a geração de códigos robustos e descartáveis, ou seja, que podem ser utilizados apenas uma única vez. Isso dificulta significativamente a clonagem e a utilização dos códigos de autenticação por usuários maliciosos. Além disso, os protocolos da S3AS podem ser considerados seguros uma vez que foram verificados formalmente utilizando a ferramenta Scyther.

Em questões de desempenho, os experimentos mostraram que o protocolo possui latência satisfatória para todos os casos de avaliação. As avaliações *on-line* foram as mais custosas com 54,91ms e 54,96ms, mas ainda assim mantiveram-se abaixo de 100ms. Este desempenho também se manteve ao utilizar OTACs em múltiplas catracas, no qual as médias para as três catracas simuladas permaneceram parecidas com 5,60ms, 5,53ns e 5,57ms, respectivamente.

Para trabalhos futuros, podem ser estudadas formas de detecção de dispositivos comprometidos ou furtados através do comportamento do usuário, utilizando aprendizagem de máquina. Pesquisas também podem ser aplicadas para evitar a reutilização de códigos, problema encontrado na autenticação *off-line*.

REFERÊNCIAS

AURESIDE. **Automação Residencial: demanda na Construção Civil**. 2019. Disponível em: <http://www.aureside.org.br/noticias/automacao-residencial--demanda-na-construcao-civil>. Citado na página 31.

Banco do Brasil. **Conheça o BB Code**. 2019. Disponível em: <https://www.bb.com.br/pbb/pagina-inicial/bb-code/pra-voce#/>. Citado na página 25.

BITTENCOURT, L. et al. The internet of things, fog and cloud continuum: Integration and challenges. **Internet of Things**, v. 3-4, p. 134 – 155, 2018. ISSN 2542-6605. Disponível em: <http://www.sciencedirect.com/science/article/pii/S2542660518300635>. Citado na página 21.

CREMERS, C. J. F. **Scyther: Semantics and verification of security protocols**. [S.l.]: Eindhoven University of Technology Eindhoven, 2006. ISBN 90-386-0804-7. Citado na página 33.

CRISTOFARO, E. D. et al. Two-factor or not two-factor? A comparative usability study of two-factor authentication. **CoRR**, abs/1309.5344, 2013. Disponível em: <http://arxiv.org/abs/1309.5344>. Citado na página 21.

CURADO, M. et al. Internet of things. In: _____. **Cyber Resilience of Systems and Networks**. Springer International Publishing, 2019. p. 381–401. ISBN 978-3-319-77492-3. Disponível em: <https://doi.org/10.1007/978-3-319-77492-3_16>. Citado na página 21.

Eldefrawy, M. H.; Alghathbar, K.; Khan, M. K. OTP-Based Two-Factor Authentication Using Mobile Phones. In: **8th Int. Conf. on Information Tech.: New Generations**. [S.l.: s.n.], 2011. p. 327–331. Citado 2 vezes nas páginas 47 e 48.

FERNANDES, R. et al. S3AS: uma Solucao de Autenticacao e Autorizacao atraves de Aplicativos de Smartphones. **Revista Eletrônica Argentina-Brasil de Tecnologias da Informação e Comunicação (ReABTIC)**, 2019. <https://arxiv.kreutz.xyz/reabtic2019_saas_s3as.pdf>. Citado na página 22.

FERNANDES, R. et al. SAAS: Uma Solução de Autenticação para Aplicativos de Smartphones. In: **4o Workshop Regional de Segurança da Informação e de Sistemas Computacionais**. Alegrete-RS, Brasil: [s.n.], 2019. <http://errc.sbc.org.br/2019/wrseg/papers/fernandes2019saas.pdf>. Disponível em: <http://errc.sbc.org.br/2019/wrseg/papers/fernandes2019saas.pdf>. Citado na página 22.

FERRAG, M. A. et al. Authentication schemes for smart mobile devices: Threat models, countermeasures, and open research issues. **CoRR**, abs/1803.10281, 2018. Disponível em: <http://arxiv.org/abs/1803.10281>. Citado na página 21.

FGV. **Pesquisa Anual do Uso de TI**. 2019. <http://bit.do/fgv-sp>. Disponível em: <https://eaesp.fgv.br/ensinoeconhecimento/centros/cia/pesquisa>. Citado na página 21.

G1. **PF investiga fraudes de R$ 3 milhões em restaurantes universitários no RS**. 2016. Disponível em: <http://g1.globo.com/rs/rio-grande-do-sul/noticia/2016/10/

pf-investiga-fraudes-de-r-3-milhoes-em-restaurantes-universitarios-no-rs.html>. Citado na página 44.

GLOBALSIGN. **What is public-key cryptography?** 2019. Disponível em: <https://www.globalsign.com/en/ssl-information-center/what-is-public-key-cryptography/>. Citado na página 47.

GOI e. **Carregamento de Saldos**. 2019. Disponível em: <https://www.e-goi.com/br/precos/carregamento-de-saldos/>. Citado na página 24.

GOOGLE. **New research: How effective is basic account hygiene at preventing hijacking**. 2019. Disponível em: <https://security.googleblog.com/2019/05/new-research-how-effective-is-basic.html>. Citado na página 24.

GUZMAN, L. B. de; SISON, A. M.; MEDINA, R. P. Md5 secured cryptographic hash value. In: **Proceedings of the 2018 International Conference on Machine Learning and Machine Intelligence**. New York, NY, USA: ACM, 2018. (MLMI2018), p. 54–59. ISBN 978-1-4503-6556-7. Disponível em: <http://doi.acm.org/10.1145/3278312.3278317>. Citado na página 45.

JENUARIO, T. et al. Verificação Automática de Protocolos de Segurança com a ferramenta Scyther. In: **4o Workshop Regional de Segurança da Informação e de Sistemas Computacionais**. Alegrete-RS, Brasil: [s.n.], 2019. <http://errc.sbc.org.br/2019/wrseg/papers/jenuario2019verificao.pdf>. Disponível em: <http://errc.sbc.org.br/2019/wrseg/papers/jenuario2019verificao.pdf>. Citado na página 33.

JENUARIO, T. et al. Verificação Automática dos Protocolos de Segurança Needham-Schroeder, WMF e CSA com a ferramenta Scyther. **Revista Eletrônica Argentina-Brasil de Tecnologias da Informação e da Comunicação (ReABTIC)**, p. –, 2019. ISSN 2446-7634. <https://arxiv.kreutz.xyz/reabtic2019_vformal_Scyther.pdf>. Citado na página 33.

KAUR, N.; DEVGAN, M.; BHUSHAN, S. Robust login authentication using time-based OTP through secure tunnel. In: IEEE. **3rd Int. Conf. on Comp. for Sustainable Global Development**. [S.l.], 2016. p. 3222–3226. Citado na página 21.

KHAMIS, M. et al. GTmoPass: Two-factor Authentication on Public Displays Using Gaze-touch Passwords and Personal Mobile Devices. In: **6th ACM International Symposium on Pervasive Displays**. New York, NY, USA: ACM, 2017. p. 8:1–8:9. ISBN 978-1-4503-5045-7. Disponível em: <http://doi.acm.org/10.1145/3078810.3078815>. Citado na página 48.

KREUTZ, D. et al. A Cyber-resilient Architecture for Critical Security Services. **J. Netw. Comput. Appl.**, Academic Press Ltd., London, UK, UK, v. 63, n. C, p. 173–189, mar. 2016. ISSN 1084-8045. Disponível em: <https://doi.org/10.1016/j.jnca.2015.09.014>. Citado na página 39.

Kreutz, D. et al. The kiss principle in software-defined networking: A framework for secure communications. **IEEE Security Privacy**, v. 16, n. 5, p. 60–70, Sep. 2018. ISSN 1540-7993. Citado na página 30.

Lara-Nino, C. A.; Morales-Sandoval, M.; Diaz-Perez, A. Small lightweight hash functions in fpga. In: **2018 IEEE 9th Latin American Symposium on Circuits Systems (LASCAS)**. [S.l.: s.n.], 2018. p. 1–4. ISSN 2473-4667. Citado na página 45.

LEE, Y. S. et al. Online banking authentication system using mobile-OTP with QR-code. In: **5th Int. Conf. on Comp. Sciences and Convergence Information Technology**. [S.l.: s.n.], 2010. p. 644–648. Citado na página 21.

MACHADO, R. et al. **Vazamentos de Dados: Histórico, Impacto Socioeconômico e as Novas Leis de Proteção de Dados**. 2019. <http://arxiv.kreutz.xyz/reabtic2019_data_leaks_ev1.pdf>. Citado na página 21.

MACHADO, R. et al. Vazamentos de Dados: Histórico, Impacto Socioeconômico e as Novas Leis de Proteção de Dados. **Revista Eletrônica Argentina-Brasil de Tecnologias da Informação e Comunicação (ReABTIC)**, 2019. <https://arxiv.kreutz.xyz/reabtic2019_data_leaks.pdf>. Citado na página 23.

MALIKI, T. E.; SEIGNEUR, J. A Survey of User-centric Identity Management Technologies. In: **The Int. Conf. on Emerging Security Information, Systems, and Technologies**. [S.l.: s.n.], 2007. p. 12–17. ISSN 2162-2108. Citado na página 21.

Olhar Digital. **Falha de segurança expõe dados biométricos de 1 milhão de pessoas**. 2019. Disponível em: <https://olhardigital.com.br/noticia/violacao-encontrada-no-sistema-da-biostar2-divulga-dados-biometricos/89194>. Citado na página 44.

Perco. **TURNSTILES GATES ACCESS CONTROL SYSTEMS**. 2017. Disponível em: <https://es.perco.com/download/catalogues-booklets/eng/Turnstiles_Catalog_eng.pdf>. Citado na página 44.

PIETRO, R. D.; ME, G.; STRANGIO, M. A. A two-factor mobile authentication scheme for secure financial transactions. In: IEEE. **International Conference on Mobile Business (ICMB'05)**. [S.l.], 2005. p. 28–34. Citado na página 21.

Pratama, A.; Prima, E. 2FMA-NetBank: A proposed two factor and mutual authentication scheme for efficient and secure internet banking. In: **8th Int. Conf. on Information Tech. and Electrical Eng.** [S.l.: s.n.], 2016. p. 1–4. Citado 2 vezes nas páginas 47 e 48.

Putra, D. S. K.; Sadikin, M. A.; Windarta, S. S-Mbank: Secure mobile banking authentication scheme using signcryption, pair based text authentication, and contactless smart card. In: **15th Int. Conf. on Quality in Research (QiR)**. [S.l.: s.n.], 2017. p. 230–234. Citado 2 vezes nas páginas 47 e 48.

SENDPULSE. **Valores para SMS**. 2019. Disponível em: <https://sendpulse.com/br/prices/sms>. Citado na página 24.

SESC/RS. **SESC/RS lança aplicativo para acesso ao Cartão Virtual**. 2018. Disponível em: <https://www.sesc-rs.com.br/noticias/sescrs-lanca-aplicativo-para-acesso-ao-cartao-virtual/>. Citado na página 21.

SHOWMETECH. **Automação com Google Home: como modernizar sua casa**. 2019. Disponível em: <https://www.showmetech.com.br/automacao-com-google-home-como-modernizar-sua-casa/>. Citado na página 31.

SLSU. **Technical Specifications**. 2018. Disponível em: <https://www.slsuonline.edu.ph/attachments/article/607/Technical%20Specifications.pdf>. Citado na página 44.

Starnberger, G.; Froihofer, L.; Goeschka, K. M. Qr-tan: Secure mobile transaction authentication. In: **2009 International Conference on Availability, Reliability and Security**. [S.l.: s.n.], 2009. p. 578–583. Citado 2 vezes nas páginas 21 e 47.

Tecmundo. **Com R$ 900 já dá para ter uma 'fechadura NFC' instalada em casa**. 2015. Disponível em: <https://www.tecmundo.com.br/seguranca/76962-r-900-ter-fechadura-nfc-instalada-casa.htm>. Citado na página 31.

Tecmundo. **Vazamento de dados biométricos expõe 130 milhões a risco de fraude na Índia**. 2015. Disponível em: <https://www.tecmundo.com.br/seguranca-de-dados/116338-vazamento-dados-biometricos-expoe-130-milhoes-risco-fraude-india.htm>. Citado na página 44.

TOTALVOICE. **Preços - TotalVoice**. 2019. Disponível em: <https://www.totalvoice.com.br/precos/?utm_source=apigeral&utm_medium=cpc&utm_campaign=google&gclid=EAIaIQobChMI0KKq0rm65gIVhw6RCh2ynAeGEAAYAyAAEgL33vD_BwE>. Citado na página 24.

Wu, L. et al. Secure key agreement and key protection for mobile device user authentication. **IEEE Transactions on Information Forensics and Security**, v. 14, n. 2, p. 319–330, Feb 2019. ISSN 1556-6013. Citado na página 21.

ZDNET. **Microsoft: Using multi-factor authentication blocks 99.9% of account hacks**. 2019. Disponível em: <https://www.zdnet.com/article/microsoft-using-multi-factor-authentication-blocks-99-9-of-account-hacks/>. Citado 2 vezes nas páginas 23 e 24.

Apêndices

APÊNDICE A – REPOSITÓRIO UTILIZADO

Clone o repositório dos experimentos pelo link:

```
$ git clone https://github.com/faelsfernandes/tcc-idgital-auth/
```

ÍNDICE