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

Rafael Fernandes, Diego Kreutz, Giulliano Paz, Rodrigo Mansilha[1]
Tadeu Jenuario[1], Roger Immich[2]

[1]Laboratório de Estudos Avançados (LEA)
Mestrado Profissional em Engenharia de Software (MPES)
Universidade Federal do Pampa (UNIPAMPA)

[2] Universidade Federal do Rio Grande do Norte (UFRN)

{faelsfernandes,giulliano94,gnoario}@gmail.com, kreutz@acm.org

rodrigomansilha@unipampa.edu.br, roger@imd.ufrn.br

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.*

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*

*Este artigo é uma versão estendida do paper [Fernandes et al. 2019], publicado no WRSeg 2019 (https://errc.sbc.org.br/2019/wrseg/) e selecionado entre os melhores trabalhos do evento para publicação em revista.

a geração de códigos de autenticação descartáveis (OTACs). Ambos os protocolos foram formalmente verificados utilizando a ferramenta de verificação automática Scyther. Os resultados demonstram 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.

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 incipientes, 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 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 QR Code estático contendo o número do Cadastro de Pessoas Físicas (CPF). 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, credenciais de usuários ainda continuam sendo o principal alvo de hackers e o principal meio que leva a vazamentos de dados [Machado et al. 2019].

Buscando mitigar algumas vulnerabilidades de segurança, foram propostos protocolos de autenticação de múltiplos fatores [Di Pietro et al. 2005, Maliki and Seigneur 2007, Starnberger et al. 2009, Lee et al. 2010, Kaur et al. 2016, Ferrag et al. 2018, Wu et al. 2019]. Uma autenticação de dois 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 serviço de mensagens curtas (*Short Message Service*, SMS), como segundo fator. Entretanto, a autenticação de múltiplos fatores leva a diversas questões e desafios de usabilidade [Cristofaro et al. 2013].

Por exemplo, os usuários do cartão virtual do SESC-RS desejam passar rapidamente pela catraca para acessar as dependências da instituição, como as academias, quadras poliesportivas, ginásios, parque aquático, entre outros. Nesse caso, o fator adicional

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

aumentaria o atraso, o que é indesejável e, talvez por isso, o mecanismo não seja disponibilizado no aplicativo da instituição. A principal questão 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*, denominada S3AS. A principal finalidade do 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* (OTACs). 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, i.e., 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 um caso de uso para demonstrar a aplicação prática da solução; (4) implementação de um protótipo da S3AS que simula a utilização da solução em catracas eletrônicas através de QR Code; e (5) verificação formal dos protocolos da solução utilizando a ferramenta Scyther.

O restante deste trabalho está organizado conforme descrito a seguir. Na Seção 2 são discutidos os protocolos de registro e autenticação propostos. Detalhes sobre o caso de uso e os principais desafios são apresentados na Seção 3. A verificação forma da solução é apresentada na Seção 4 e os trabalhos relacionados na Seção 5. Por fim, a Seção 6 apresenta as considerações finais, conclusão e trabalhos futuros.

2. Os Protocolos da S3AS

A S3AS é composta por dois protocolos: identificação, onde o aplicativo do usuário é vinculado a um único dispositivo móvel, e verificação, onde são gerados códigos descartáveis de autenticação.

2.1. 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 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 Protocolo 1.

O processo de registro inicia com uma conexão utilizando o *Transport Layer Security* (TLS) entre o aplicativo e o servidor (linha 1). Na sequência, linhas 2 à 4, o servidor envia três códigos distintos, utilizando o canal TLS e dois canais *out-of-band*, neste caso

Protocolo 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	$K_{T1} \leftarrow$ H($tls\_session\_key \| \| code_1 \| \| code_2 \| \| code_3$)
6. Cliente $\rightarrow$ Servidor	[Cliente, *nonce*, $E_{K_{T1}}$(IMEI, app_rand1)], $HMAC_{K_{T1}}$
7. Cliente, Servidor	$K_{T2} \leftarrow$ H($IMEI \| \| app\_rand1 \| \| K_{T1}$)
8. Servidor $\rightarrow$ Cliente	[Servidor, *nonce*, $E_{K_{T2}}(server\_rand)$], $HMAC_{K_{T2}}$
9. Cliente, Servidor	$K_m \leftarrow$ H($K_{T1} \| \| K_{T2} \| \| IMEI \| \| app\_rand1 \| \| server\_rand$)
10. Cliente $\rightarrow$ Servidor	[Cliente, V_MKEY, *nonce*, $E_{K_m}(app\_rand2)$], $HMAC_{K_m}$
11. Servidor $\rightarrow$ Cliente	[Servidor, V_MKEY, *nonce*, $E_{K_m}(app\_rand2 + 1)$], $HMAC_{K_m}$

específico 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.

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.

Enquanto o IMEI e a chave mestra estiverem válidos ou em uso pelo usuário, não é possível registrar outro dispositivo utilizando as mesmas credencias. O usuário pode utilizar apenas um dispositivo por vez. Para usar um novo dispositivo, o usuário deve primeiro revogar o aplicativo/registro atual. Esse procedimento de segurança é adotado

por diferentes bancos internacionais e nacionais, como o Revolut[2], N26[3], Banco Inter[4] e NuBank[5].

2.2. 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 demonstrado 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[6]. A chave inicial de geração dos códigos únicos de autenticação pode ser tão simples quanto K_c = H($K_m||K_c$). Como a chave K_c inicia vazia, a primeira K_c é igual a 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, isto é, K_c = H($K_m||K_c$). Já os códigos OTAC são gerados da seguinte forma: OTAC = H^N(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 OTAC passados a partir do código OTAC atual. Essa propriedade pode ser assegurada através da propriedade de irreversibilidade das funções de *hash* criptográfica.

Para exemplificar a utilização dos códigos OTAC, 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 (assinatura da mensagem) 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.

3. Caso de uso e desafios

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 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

[2]`https://www.revolut.com`
[3]`https://n26.com`
[4]`https://www.bancointer.com.br`
[5]`https://nubank.com.br/`
[6]`https://csrc.nist.gov/projects/hash-functions`

gera automaticamente um novo QR Code contendo o novo OTAC de autenticação, como pode ser observado nas linhas 1 e 2 do Protocolo 2.

Protocolo 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 $\leftarrow$H^{iA-iS}(OTAC)
6.	Verifica HMAC utilizando o OTAC como chave

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 totalmente 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 Codes.

Vale ressaltar que códigos únicos, com alta entropia, como os OTACs, são mais confiáveis do que dados biométricos, que são estáticos por natureza. Em outras palavras, dados biométricos são iguais a uma senha que você não pode trocar. Se a senha estática 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.

4. 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. 2019b, Jenuario et al. 2019a].

4.1. 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*

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 }
```

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 $code1$, $code2$ e $code3$. A chave K_{t1} dos agentes `Cliente` e `Servidor` é gerada nas linhas 16 e 17, respectivamente, garantindo que ela seja igual em ambos os agentes.

As variáveis $nonce$, $IMEI$, $appRand1$ e $appRand2$ (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 $code1$, $code2$, $code3$ 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 [Jenuario et al. 2019b, Jenuario et al. 2019a]. Para o caso do código do algoritmo de identificação, como pode ser observado na Figura 1, 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.

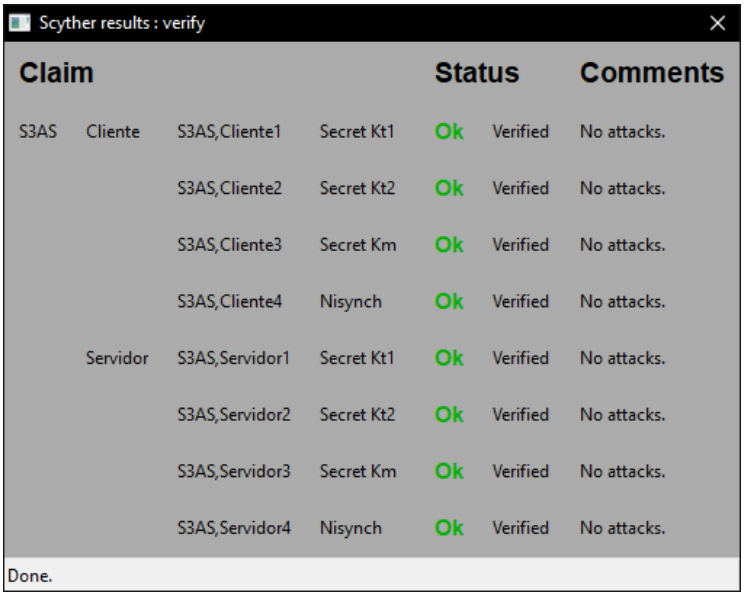

Figura 1. Verificação do Protocolo de Identificação (Algoritmo 1)

4.2. 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 3. 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 2 apresenta o resultado da análise da Scyther. Como pode ser observado,

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 }
```

a análise retornou *Ok* (no *Status*) para as afirmações do usuário e da catraca, indicando que os OTACs estão seguros.

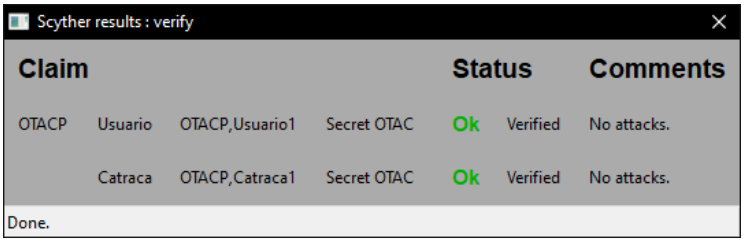

Figura 2. Verificação do protocolo de autenticação

5. Soluções de identificação e verificação

Tabela 1. Soluções de identificação e verificação robustas

Solução	**Projetada para**	**Autenticação**	***Canal(is) out-of-band***
[Eldefrawy et al. 2011]	*Internet Banking*	Usuário/senha e OTP via aplicação móvel	OTP é enviado para dispositivo móvel
[Pratama and 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 et al. 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

A Tabela 1 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.

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 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.

6. Conclusão

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.

Referências

Bittencourt, L., Immich, R., Sakellariou, R., Fonseca, N., Madeira, E., Curado, M., Villas, L., DaSilva, L., Lee, C., and Rana, O. (2018). The internet of things, fog and cloud continuum: Integration and challenges. *Internet of Things*, 3-4:134 – 155.

Cremers, C. J. F. (2006). *Scyther: Semantics and verification of security protocols*. Eindhoven University of Technology Eindhoven.

Cristofaro, E. D., Du, H., Freudiger, J., and Norcie, G. (2013). Two-factor or not two-factor? A comparative usability study of two-factor authentication. *CoRR*, abs/1309.5344.

Curado, M., Madeira, H., da Cunha, P. R., Cabral, B., Abreu, D. P., Barata, J., Roque, L., and Immich, R. (2019). *Internet of Things*, pages 381–401. Springer International Publishing.

Di Pietro, R., Me, G., and Strangio, M. A. (2005). A two-factor mobile authentication scheme for secure financial transactions. In *International Conference on Mobile Business (ICMB'05)*, pages 28–34. IEEE.

Eldefrawy, M. H., Alghathbar, K., and Khan, M. K. (2011). OTP-Based Two-Factor Authentication Using Mobile Phones. In *8th Int. Conf. on Information Tech.: New Generations*, pages 327–331.

Fernandes, R., Paz, G., Kreutz, D., Mansilha, R., and Immich, R. (2019). SAAS: Uma Solução de Autenticação para Aplicativos de Smartphones. In *4o Workshop Regional de Segurança da Informação e de Sistemas Computacionais*, Alegrete-RS, Brasil. `http://errc.sbc.org.br/2019/wrseg/papers/fernandes2019saas.pdf`.

Ferrag, M. A., Maglaras, L. A., Derhab, A., Vasilakos, A. V., Rallis, S., and Janicke, H. (2018). Authentication schemes for smart mobile devices: Threat models, countermeasures, and open research issues. *CoRR*, abs/1803.10281.

FGV (2019). Pesquisa Anual do Uso de TI. `http://bit.do/fgv-sp`.

Jenuario, T., Chervinski, J. O., Paz, G., Fernandes, R., Beltran, R., and Kreutz, D. (2019a). 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)*, pages –. `https://arxiv.kreutz.xyz/reabtic2019_vformal_Scyther.pdf`.

Jenuario, T., Chervinski, J. O., Paz, G., and Kreutz, D. (2019b). 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. `http://errc.sbc.org.br/2019/wrseg/papers/jenuario2019verificao.pdf`.

Kaur, N., Devgan, M., and Bhushan, S. (2016). Robust login authentication using time-based OTP through secure tunnel. In *3rd Int. Conf. on Comp. for Sustainable Global Development*, pages 3222–3226. IEEE.

Khamis, M., Hasholzner, R., Bulling, A., and Alt, F. (2017). GTmoPass: Two-factor Authentication on Public Displays Using Gaze-touch Passwords and Personal Mobile Devices. In *6th ACM International Symposium on Pervasive Displays*, pages 8:1–8:9, New York, NY, USA. ACM.

Lee, Y. S., Kim, N. H., Lim, H., Jo, H., and Lee, H. J. (2010). Online banking authentication system using mobile-OTP with QR-code. In *5th Int. Conf. on Comp. Sciences and Convergence Information Technology*, pages 644–648.

Machado, R., Kreutz, D., Paz, G., and Rodrigues, G. (2019). Vazamentos de Dados: Histórico, Impacto Socioeconômico e as Novas Leis de Proteção de Dados. `http://arxiv.kreutz.xyz/reabtic2019_data_leaks_ev1.pdf`.

Maliki, T. E. and Seigneur, J. (2007). A Survey of User-centric Identity Management Technologies. In *The Int. Conf. on Emerging Security Information, Systems, and Technologies*, pages 12–17.

Pratama, A. and Prima, E. (2016). 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.*, pages 1–4.

Putra, D. S. K., Sadikin, M. A., and Windarta, S. (2017). 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)*, pages 230–234.

SESC/RS (2018). SESC/RS lança aplicativo para acesso ao Cartão Virtual. `http://bit.do/sesc-rs`.

Starnberger, G., Froihofer, L., and Goeschka, K. M. (2009). Qr-tan: Secure mobile transaction authentication. In *2009 International Conference on Availability, Reliability and Security*, pages 578–583.

Wu, L., Wang, J., Choo, K. R., and He, D. (2019). Secure key agreement and key protection for mobile device user authentication. *IEEE Transactions on Information Forensics and Security*, 14(2):319–330.