Identifica ISP: Autenticação Mútua entre Múltiplas Entidades para Serviços de Suporte Técnico Prestados por ISPs

Vagner E. Quincozes[1], Silvio E. Quincozes[2], Diego Kreutz[1], Rodrigo B. Mansilha[1]

[1] Universidade Federal do Pampa - UNIPAMPA

[2]Universidade Federal Fluminense - UFF

vagnerquincozes.aluno@unipampa.edu.br, sequincozes@id.uff.br, diegokreutz@unipampa.edu.br, mansilha@unipampa.edu.br

Resumo. *Os Internet Service Providers (ISPs) ainda utilizam mecanismos frágeis para autenticação e identificação entre as entidades envolvidas em suporte técnico especializado (i.e., clientes, técnicos e gestores). Tais mecanismos geram insegurança aos usuários, pois costumam adotar dados estáticos não autenticáveis (e.g., CPF em cartões físicos ou virtuais) que podem ser facilmente roubados, clonados ou reproduzidos. Neste trabalho, propomos um sistema para autenticação e identificação de clientes, técnicos e gestores de ISP, composto por: (i) um aplicativo para dispositivos móveis suportado por um sistema de back-end e (ii) a implementação de protocolos de segurança que oferecem serviços de identificação e autenticação. Uma demonstração da aplicação ilustra o modo de utilização do sistema. Uma avaliação conceitual dos serviços de segurança, amparada pela análise dos protocolos por meio da ferramenta Scyther, sustenta a segurança provida pelo sistema.*

1. Introdução

O aumento da complexidade e quantidade das *Local Area Networks* (LANs) tem ampliado a dependabilidade dessas redes e a demanda por suporte técnico especializado relacionado. Uma parte dessa demanda, que pode ser motivada por problemas externos ou internos às LANs, é encaminhada para os *Internet Service Providers* (ISPs). As entidades envolvidas em suporte técnico no âmbito dos ISPs ainda utilizam protocolos e mecanismos frágeis de autenticação. Empiricamente, observamos no estado do Rio Grande do Sul que os ISPs regionais empregam cartões físicos e/ou virtuais, baseados em dados estáticos não autenticáveis, que podem ser facilmente roubados, reproduzidos ou clonados. As operadoras (*i.e.,* ISPs de escala nacional) empregam fatores adicionais de segurança, como ligações telefônicas, que incrementam a segurança, mas impactam negativamente na experiência de usuário e nos custos operacionais. Diante desse cenário, *argumentamos que prover algum mecanismo ágil e robusto para autenticação mútua entre entidades envolvidas em serviço de suporte técnico torna-se um problema chave.*

Neste trabalho propomos o Identifica ISP: um sistema de segurança para prover autenticação mútua entre múltiplas entidades envolvidas em prestação de serviços com foco em suporte técnico realizados por ISPs. A arquitetura do sistema é organizada em camadas. Essa organização permite atingir diferentes níveis de compromisso entre segurança e usabilidade impactando minimamente no restante do sistema. Essa organização também permite a manutenção de módulos específicos a cada escala de negócio (*e.g.*, operadora nacional ou ISP regional), e a cada paradigma tecnológico, atual

(*i.e.*, humanos) e futuro (*e.g.*, sistemas especialistas, robôs (semi) automatizados). Ademais, o sistema oferece autenticação mútua para uma quantidade arbitrária de entidades, que pode variar conforme o tipo de suporte técnico prestado (*e.g.*, configuração de uma rede doméstica ou correção de falha em uma rede organizacional). Em síntese, este trabalho apresenta as seguintes contribuições técnicas:

- **(a)** especificação de uma arquitetura para prover autenticação mútua ao suporte técnico prestado por ISPs;
- **(b)** uma implementação da arquitetura voltada para suporte técnico atualmente prestado por ISPs, denominada Identifica ISP;
- **(c)** uma demonstração do Identifica ISP e uma avaliação conceitual sobre serviços de segurança oferecidos pelo sistema.

O restante deste trabalho está organizado como segue. A arquitetura e a implementação são apresentadas, respectivamente nas seções 2 e 3. Uma demonstração e uma avaliação do protocolo são apresentadas na Seção 4. Por fim, considerações finais são tecidas na Seção 5.

2. Arquitetura

A arquitetura é organizada em camadas e módulos para que possa ser moldada de forma a impactar minimamente no restante do sistema. Essa flexibilidade pode ser utilizada principalmente para atingir diferentes níveis de compromisso entre segurança e usabilidade dependendo do contexto. Além disso, a flexibilidade pode ser aproveitada para gerenciar módulos específicos a cada escala de negócio. Por exemplo, uma operadora nacional tende a atender um público mais heterogêneo que um ISP regional, cenário no qual algumas simplificações potencialmente poderão ser úteis. A flexibilidade também permite acomodar evoluções tecnológicas. Muitas atividades de suporte técnico que atualmente são realizadas por humanos, no futuro poderão ser executadas de maneira (semi) automatizada por robôs.

A Figura 1 resume a arquitetura do sistema proposto. Na figura, as colunas da esquerda e do centro apresentam, respectivamente, as camadas e exemplos de módulos de cada camada, conforme será explicado a seguir. A coluna da direita mostra a instância implementada, que será detalhada na próxima seção.

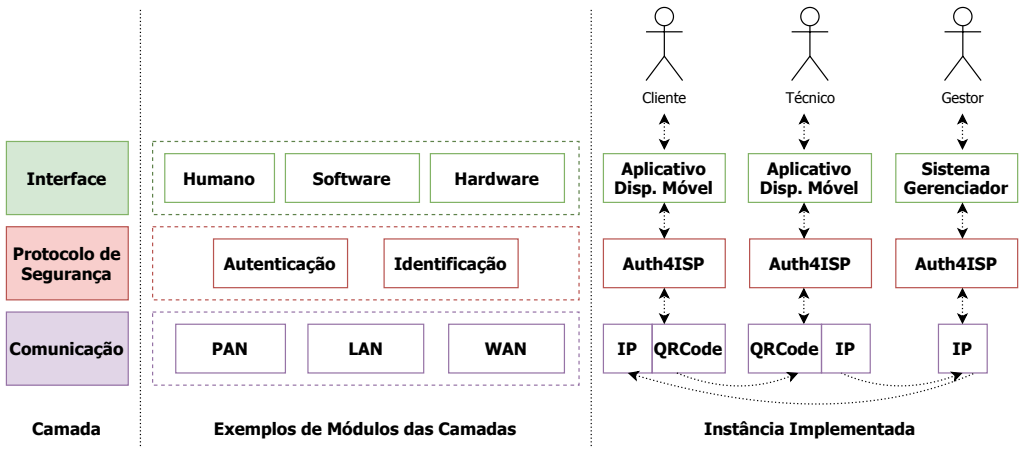


Figura 1. Arquitetura do sistema proposto

Definimos três camadas principais: (*i*) Interface, (*ii*) Protocolos de Segurança e (*iii*) Comunicação. A primeira camada reúne os módulos necessários para realizar uma **interface** entre um conjunto potencialmente variado de entidades - presentes e futuras - que possam ser envolvidas no serviço, sejam elas humanas ou tecnológicas, com o restante do sistema. Por exemplo, entidades humanas podem ser acopladas ao sistema através de um aplicativo. O cliente e o técnico podem preferir um aplicativo para dispositivo móvel e minimalista, que essencialmente permita a um cliente autenticar um técnico e vice-versa. Em contraste, um gestor poderia preferir uma interface relativamente mais complexa, que contenha um leque de funcionalidades relativamente maior (*e.g.*, gerenciamento de clientes, técnicos e atendimentos). Uma entidade tecnológica de hardware pode ser, por exemplo, o Gateway de Acesso Controlado (GAC) [Torres et al. 2020], que permite gerenciamento remoto seguro e padronizado à LAN com baixo custo. Uma entidade tecnológica de software pode ser, por exemplo, um sistema de monitoramento.

A segunda camada reúne **protocolos de segurança**, que oferecerem serviços de identificação e autenticação. O conjunto de protocolos pode ser moldado para atender variadas quantidades de entidades envolvidas e compromissos entre segurança e eficiência. É importante observar que a quantidade de entidades envolvidas em um processo pode variar dependendo da situação. Por exemplo, um atendimento pode envolver apenas um cliente doméstico, um técnico e um gestor. Outro atendimento pode envolver um número maior de responsáveis pelo cliente, como no caso de uma organização, equipamentos, profissionais e gestores responsáveis pelo ISP.

A terceira camada representa a infraestrutura subjacente necessária para realizar a **comunicação** entre as entidades. Um exemplo é a pilha de protocolos TCP/IP, que permite a execução dos protocolos através de uma rede local ou da Internet. Outros exemplos são as tecnologias de redes de área pessoal, que podem ser utilizadas para transmitir dados do processo de identificação e autenticação (*e.g.*, QR Code, RFID).

3. Implementação

A instância implementada é voltada para humanos pois acreditamos que esta seja a principal demanda dos ISPs atualmente. O objetivo é prover autenticação de entidades que foram identificadas previamente, para mitigar ameaças relacionadas a personificação. Assumimos que usuários autênticos sejam honestos. Consideramos um cenário particular no qual um cliente recebe a visita de um técnico terceirizado pelo seu ISP. É importante ressaltar que um atendimento técnico poderia envolver uma quantidade de entidades maior (*e.g.*, equipe de infraestrutura) ou menor (*e.g.*, técnico interno do ISP).

Na coluna da direita na Figura 1 observe-se três classes de entidade humanas: cliente, técnico e gestor. Essas três classes interagem com o sistema principalmente através de um aplicativo para dispositivos móveis. Na camada intermediária, implementamos um conjunto de protocolos de segurança que propusemos anteriormente [Quincozes et al. 2020]. Na camada de comunicação adotamos, dependendo do contexto, QR Code ou a pilha TCP/IP. As duas subseções seguintes apresentam, respectivamente, as implementações das camadas de interface e de protocolo de segurança.

3.1. Interfaces Humano-Computador

A Figura 2 apresenta as tecnologias de software utilizadas na implementação das interfaces. Essas tecnologias podem ser organizadas em duas camadas: *Front-end* de Interface

e *Back-end* de Interface. A camada *Front-end* inclui um aplicativo móvel, denominado Identifica ISP, (do tipo .apk), um Gestor Automatizado, responsável por autenticar as entidades humanas, desenvolvido em Java (.jar), e; uma API de Integração para obter dados cadastrais e outros sistemas (*e.g.*, CRM, ERP) e facilitar o processo de manutenção de cadastro de usuários. Por fim, a camada *Back-end* de Interface é responsável por manter as informações dos usuários e por prover persistência para as sessões. Assim, caso alguma entidade perca temporariamente a sua conectividade, o processo continuará de onde parou quando a conectividade for reestabelecida. Optamos por utilizar o *Firebase Realtime Database* como tecnologia de *Back-end* por ser uma solução popular para armazenar e sincronizar dados.

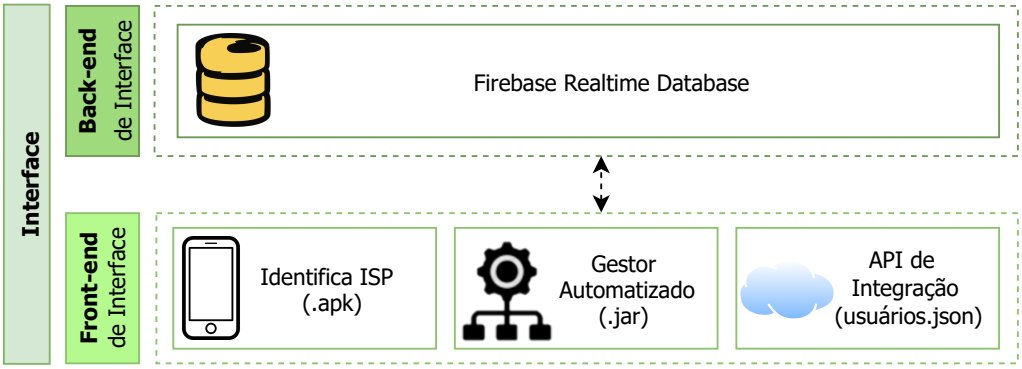


Figura 2. Interface implementada

As três principais telas do aplicativo para dispositivo móvel implementado são exemplificadas na Figura 3. A primeira tela permite ao usuário escolher entre as funções de autenticação: (*i*) gerar código de autenticação e (*ii*) ler código de autenticação. A segunda tela mostra um *QR Code* gerado dinamicamente a ser apresentado. A terceira tela exibe a identificação da(s) outra(s) entidade(s) e permite confirmar.

(a) Funcionalidades (b) QRCode (c) Identificação

Figura 3. Exemplos das principais telas do aplicativo

O processo de autenticação inicia quando uma entidade (*e.g.*, cliente) clica no botão gerar código de autenticação. A partir desse momento, o QR Code com as informações do cliente é gerado. Caso ocorra algum problema durante a execução do

protocolo, o aplicativo apresenta uma mensagem de erro. No fluxo normal, outra entidade (*e.g.*, técnico) deve clicar em ler código de autenticação e utilizar o leitor para ler o QR Code apresentado pelo cliente. Assim que o QR Code é lido, as informações são enviadas para o gestor automatizado, que verifica ambas identidades. Se as identidades forem confirmadas, a tela de identificação exibida na Figura 3(c) é apresentada ao cliente com as informações do técnico (*i.e.*, foto, nome, avaliação, etc.). De maneira semelhante, uma tela de identificação é exibida para o técnico com as informações do cliente. Ambas entidades devem verificar se as identidades apresentadas estão corretas.

Na atual implementação, a interface de Gestor é simplificada. Através dessa interface, o gestor além de autenticar usuários, pode gerenciar (*i.e.*, listar, editar, adicionar e revogar) clientes e técnicos. Na prática, esse processo deve ser agilizado tirando proveito de alguma integração com sistemas terceirizados. Para ilustrar esse cenário, a funcionalidade "adicionar usuário" implementada pergunta apenas o CPF do usuário – o restante dos dados de interesse são obtidas através de uma consulta ao *Firebase Realtime Database*. Assumimos que aplicações de terceiros ofereçam alguma API de integração, que gere, por exemplo, um arquivo em formato JSON. Para tornar a demonstração auto-contida, abstraímos a API de terceiros criando um arquivo JSON com dados sintéticos e carregando-o diretamente para o *Firebase Realtime Database*. Complementarmente, foi implementada uma interface que representa um gestor automatizado. Em resumo, essa interface roda um processo que verifica e autoriza automaticamente todas as requisições. Como trabalhos futuros, pretendemos adicionar novas configurações para o gestor, como padrões de atuação (autorizar, negar ou confirmar) personalizáveis, que variem dependendo da situação. Por exemplo, o atendimento recorrente de um técnico a um cliente em horário comercial pode ser autorizado por padrão. Em contraste, o primeiro atendimento para um cliente por um novo técnico durante um feriado pode exigir confirmação.

3.2. Sistema de Protocolos de Segurança

Foi implementada uma versão simplificada do sistema de protocolos denominado Auth4ISP [Quincozes et al. 2020], que é composto por um mecanismo de identificação e um protocolo de autenticação. O mecanismo de identificação vincula alguma pessoa a algum dispositivo físico. Ele é uma instância do protocolo de identificação do sistema Auth4App [Kreutz et al. 2020]. Especificamente, cada *Front-end* Auth4ISP representa um cliente Auth4App e o *Back-end* Auth4ISP representa o servidor Auth4App.

O protocolo de autenticação, denominado BKE-Auth4ISP, verifica se as entidades/dispositivos previamente identificadas/vinculadas são autênticas e válidas. Em suma, o BKE-Auth4ISP combina o protocolo do sistema Auth4App [Kreutz et al. 2020] com o protocolo β^* da família de protocolos proposto em [Cremers and Mauw 2006]. À exemplo dos membros dessa família, o BKE-Auth4ISP pode ser facilmente modificado para autenticar uma quantidade variada de entidades em um mesmo ciclo de autenticação. O BKE β^* foi o membro escolhido para servir de base pois é o protocolo menos computacionalmente custoso da sua família. A diferença é que o BKE-Auth4ISP usa Auth4App como mecanismo em substituição a uma infra-estrutura PKI clássica, adotada pelo protocolo original.

Para tornar este trabalho auto-contido, detalhamos o BKE-Auth4ISP no Protocolo 1. Nele, há três entidades (Cliente, Técnico e Gestor), que devem ser previamente

identificadas, e cada linha mostra uma mensagem. As colunas contém, em ordem da esquerda para direita, respectivamente: (*i*) identificador do conjunto de mensagens (*e.g.*, M1); (*ii*) entidades envolvidas e sentido da mensagem, (*iii*) descrição do conteúdo da mensagem, e (*iv*) relação com o aplicativo. Todas as mensagens são cifradas utilizando uma função de encriptação `E`, cuja chave secreta é um *One-Time Authentication Code* (OTAC) definido no protocolo de identificação. Os OTACs são definidos aos pares (*e.g.*, cliente e técnico - $E_{cli,tec} = E_{tec,cli}$). Assumimos que os OTACs podem ser inicializados e transmitidos por um canal seguro quando o ISP designa um atendimento técnico a um cliente.

O processo de autenticação inicia quando o Cliente envia M1 para o Técnico, contendo a sua identificação (`Cli`), a identificação do Gestor (`Isp`), um nonce de autenticação (`nonce_cli`) mais um `HMAC` dessas três informações. Ao receber a M1, o Técnico adiciona a sua identificação (`Tec`), um nonce de autenticação (`nonce_tec`) mais um `HMAC` e envia a M2 para o Gestor. Em seguida, o Gestor gera a M3 contendo os nonces de autenticação do Cliente (`nonce_cli`), do Técnico (`nonce_tec`) e do Gestor (`nonce_isp`), mais a identificação do Técnico (`Tec`) e a sua identificação (`Isp`) e um `HMAC` dessas informações. Ao receber a M3, o Cliente envia para o Técnico a M4 contendo o nonce de autenticação do Gestor (`nonce_isp`) e do Técnico (`nonce_tec`) mais um `HMAC`. Ao receber a M4, o Técnico envia para o Gestor a M5, composta pelo nonce do Gestor (`nonce_isp`) e um `HMAC`, finalizando o processo de autenticação.

Protocolo 1. BKE-Auth4ISP e sua integração com o aplicativo

M1	Cliente → Técnico	$[E_{cli,tec}$(Cli, Isp, nonce-cli)]HMAC	tela 3(b)
M2	Técnico → Gestor do ISP	$[E_{tec,isp}$(Tec, Cli, nonce-cli, nonce-tec)]HMAC	após ler QR-Code
M3	Gestor do ISP → Cliente	$[E_{isp,cli}$(nonce-cli, nonce-tec, nonce-isp, Tec, Isp)]HMAC	tela 3(c)
M4	Cliente → Técnico	$[E_{cli,tec}$(nonce-isp,nonce-tec)]HMAC	tela 3(c)
M5	Técnico → Gestor do ISP	$[E_{tec,isp}$(nonce-isp)]HMAC	

Na implementação do BKE-Auth4ISP, todas as mensagens trocadas pelo protocolo são cifradas antes do envio e decifradas na chegada. Para isso, adotamos o algoritmo de chave simétrica *Advanced Encryption Standard* (AES), cuja chave secreta é um OTAC. O AES opera com tamanhos de chaves de 128, 192 e 256 bits e criptografia de blocos de 128 bits [Daemen and Rijmen 2002]. Tanto o algoritmo de cifra, quanto o HMAC adotam a biblioteca *Message Digest*[1] com a instância SHA-256 para gerar funções *hash* criptográficas.

4. Avaliação

Esta seção apresenta uma demonstração da aplicação (Subseção 4.1) e uma análise sobre o protocolo de segurança (Subseção 4.2). O código-fonte do sistema, o protocolo em linguagem Scyther, e documentos complementares necessários para reproduzir os cenários estão disponíveis publicamente[2].

4.1. Demonstração da Aplicação

Foram previstas 3 maneiras para configurar o ambiente, com variados compromissos entre complexidade e abrangência dos componentes instalados. Elas exigem duas ou três

[1] `https://docs.oracle.com/javase/7/docs/api/java/security/MessageDigest.html`

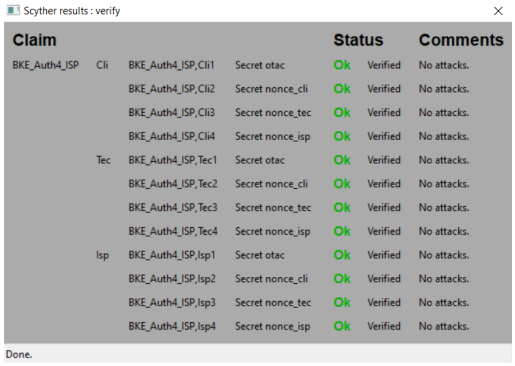
[2] `https://github.com/vagnerereno/IdentificaISP_SBSeg2021.git`

máquinas (possivelmente virtuais): duas com o sistema operacional Android e, opcionalmente, outra com *Java SE Development Kit* (JDK). Essas máquinas hospedam as aplicações Front-end e devem ter acesso à Internet de modo a possibilitar a sua conexão com o *back-end*.

A demonstração é organizada em três etapas: configuração de ambiente, execução do sistema e análise de registros. Na primeira etapa, é demonstrado como instalar o aplicativo (.apk) no ambiente Android e como executar o Gestor Automatizado (.jar) no ambiente JDK. Na etapa de execução são apresentadas duas funcionalidades principais, na seguinte ordem: *(i)* processo de autenticação, e *(ii)* adição de usuários simulando uma API externa. Na terceira e última etapa, é mostrado a saída de *log* do processo, incluindo o conteúdo das mensagens trocadas durante o processo de autenticação.

4.2. Aspectos de Segurança do Protocolo

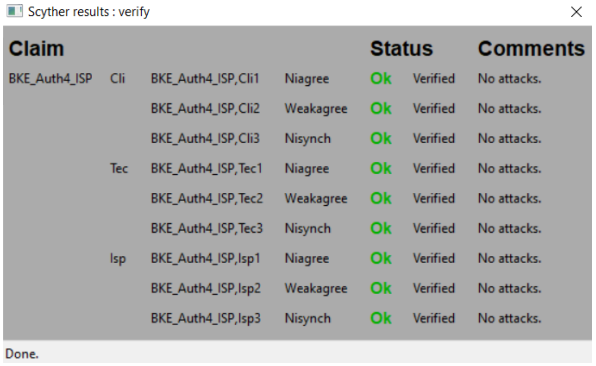
Em trabalhos anteriores, apresentamos avaliações conceituais do protocolo Auth4App [Kreutz et al. 2020] e do protocolo BKE-Auth4ISP [Quincozes et al. 2020] usando a ferramenta de verificação automática Scyther [Cremers 2008]. Essa ferramenta recebe como entrada um protocolo descrito na linguagem `spdl` e retorna como saída um relatório sobre a segurança do protocolo. Para tornar este trabalho auto-contido reproduzimos o resultado da avaliação do protocolo BKE-Auth4ISP. Na Figura 4 são apresentados os resultados sobre as propriedades secretas. Notamos que nas coluna *Status* e *Comments* as propriedades apresentam *Ok Verified* e *No attacks*, o que significa que as propriedades verificadas não apresentam possibilidades de ataques. Similarmente, a Figura 4(b) indica que as propriedades *Niagree*, *Weakagree* e *Nisynch* foram verificadas e não estão suscetíveis a ataques, conforme a metodologia adotada.


Scyther results : verify

Claim				Status		Comments
BKE_Auth4_ISP	Cli	BKE_Auth4_ISP,Cli1	Secret otac	Ok	Verified	No attacks.
		BKE_Auth4_ISP,Cli2	Secret nonce_cli	Ok	Verified	No attacks.
		BKE_Auth4_ISP,Cli3	Secret nonce_tec	Ok	Verified	No attacks.
		BKE_Auth4_ISP,Cli4	Secret nonce_isp	Ok	Verified	No attacks.
	Tec	BKE_Auth4_ISP,Tec1	Secret otac	Ok	Verified	No attacks.
		BKE_Auth4_ISP,Tec2	Secret nonce_cli	Ok	Verified	No attacks.
		BKE_Auth4_ISP,Tec3	Secret nonce_tec	Ok	Verified	No attacks.
		BKE_Auth4_ISP,Tec4	Secret nonce_isp	Ok	Verified	No attacks.
	Isp	BKE_Auth4_ISP,Isp1	Secret otac	Ok	Verified	No attacks.
		BKE_Auth4_ISP,Isp2	Secret nonce_cli	Ok	Verified	No attacks.
		BKE_Auth4_ISP,Isp3	Secret nonce_tec	Ok	Verified	No attacks.
		BKE_Auth4_ISP,Isp4	Secret nonce_isp	Ok	Verified	No attacks.

Done.

(a) Propriedades secretas.


Scyther results : verify

Claim				Status		Comments
BKE_Auth4_ISP	Cli	BKE_Auth4_ISP,Cli1	Niagree	Ok	Verified	No attacks.
		BKE_Auth4_ISP,Cli2	Weakagree	Ok	Verified	No attacks.
		BKE_Auth4_ISP,Cli3	Nisynch	Ok	Verified	No attacks.
	Tec	BKE_Auth4_ISP,Tec1	Niagree	Ok	Verified	No attacks.
		BKE_Auth4_ISP,Tec2	Weakagree	Ok	Verified	No attacks.
		BKE_Auth4_ISP,Tec3	Nisynch	Ok	Verified	No attacks.
	Isp	BKE_Auth4_ISP,Isp1	Niagree	Ok	Verified	No attacks.
		BKE_Auth4_ISP,Isp2	Weakagree	Ok	Verified	No attacks.
		BKE_Auth4_ISP,Isp3	Nisynch	Ok	Verified	No attacks.

Done.

(b) Propriedades Niagree, Weakgree, Nisynch.

Figura 4. Resultado da verificação do protocolo *BKE-Auth4ISP* com *Scyther*.

Com base na avaliação conceitual do sistema de protocolos e nas tecnologias adotadas na implementação, apresentamos a seguinte análise sobre as propriedades de segurança da informação oferecidas pelo sistema.

- **Confidencialidade.** Todas as mensagens trocadas entre o cliente, técnico e o gestor do ISP são criptografadas com uma função *encrypt*. Tal função é composta por um algoritmo de chave simétrica cuja chave secreta é um OTAC gerado pelo mecanismo de identificação.

- **Integridade e Autenticidade.** Essas duas propriedades de segurança são garantidas através do emprego de HMAC e *nonces* em cada uma das mensagens trocadas pelo protocolo. O uso de HMACs envolve uma função *hash* criptográfica e uma chave secreta. Isso permite comparar mensagens resumidas antes e depois da transmissão para garantir a integridade de cada mensagem. Ademais, cada participante gera um novo número arbitrário (*i.e.*, *nonce)* para cada execução do protocolo. A correta recepção do próprio *nonce* atesta a integridade e autenticidade da sequência de mensagens trocadas. Por exemplo, um técnico que tente forjar a identidade de um cliente não logrará êxito. O técnico não conseguirá descobrir o *nounce_isp*, a ser recebida na M3, pois ela estará encriptada com a chave compartilhada entre o Cliente e o gestor do ISP ($E_{isp,cli}$).

5. Considerações Finais

Neste trabalho implementamos uma solução de autenticação mútua entre múltiplas entidades para serviços de suporte técnico prestados por ISPs. Um repositório público (`https://github.com/vagnerereno/IdentificaISP_SBSeg2021.git`) reúne o sistema implementado, bem como o material de suporte necessário para reproduzir ambiente de testes, e a avaliação das propriedades de segurança do protocolo usando a ferramenta Scyther.

Acreditamos que a solução possa contribuir para a área de segurança da informação e de sistemas computacionais de maneiras particular e geral. Particularmente, esperamos que a solução possa ser adotada para efetivamente incrementar a segurança em serviços de suporte técnico atualmente prestados por ISPs e, consequentemente, ampliar a segurança das LANs. No âmbito geral, acreditamos que as lições apresentadas neste trabalho possam ser consideradas no projeto de sistemas de autenticação mútua entre múltiplas entidades voltados para serviços de suporte técnicos prestados por ISPs no futuro e/ou serviços de suporte técnico prestados por outras indústrias.

Referências

Cremers, C. and Mauw, S. (2006). A Family of Multi-Party Authentication Protocols. In *First Benelux Workshop on Information and System Security (WISSec)*.

Cremers, C. J. (2008). The Scyther Tool: Verification, Falsification, and Analysis of Security Protocols. In *International conference on computer aided verification*, pages 414–418. Springer.

Daemen, J. and Rijmen, V. (2002). *The design of Rijndael*, volume 2. Springer.

Kreutz, D., Fernandes, R., Paz, G., Jenuario, T., Mansilha, R., Immich, R., and Miers, C. C. (2020). Auth4App: Protocols for Identification and Authentication using Mobile Applications. In *SBC 20th International Brazilian Symposium on Information and Computational Systems Security (SBSeg)*, pages 1–14. SBC.

Quincozes, V. E., Temp, D., Quincozes, S. E., Kreutz, D., and Mansilha, R. B. (2020). Sistema para Autenticação entre Clientes, Técnicos e ISPs. In *5o Workshop Regional de Segurança da Informação e de Sistemas Computacionais*, Joinville-SC, Brasil.

Torres, R., Quincozes, V. E., Mansilha, R. B., and Kreutz, D. (2020). Gateway de Acesso Controlado-GAC. *Anais do Salão Internacional de Ensino, Pesquisa e Extensão*, 12(2).