UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE COMPUTAÇÃO

Giann Carlos Spilere Nandi

VERIFICAÇÃO FORMAL DE PROTOCOLOS DE SEGURANÇA VOLTADOS À REDES DE SENSORES SEM FIO

Araranguá

2017

Giann Carlos Spilere Nandi

VERIFICAÇÃO FORMAL DE PROTOCOLOS DE SEGURANÇA VOLTADOS À REDES DE SENSORES SEM FIO

Monografia submetida ao Curso de Engenharia de Computação para a obtenção do Grau de Bacharelado em Engenharia de Computação.
Orientadora: Prof. Dr. Analúcia Schiaffino Morales

Araranguá

2017

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Nandi, Giann Carlos Spilere
Verificação formal de protocolos de segurança voltados à redes de sensores sem fio / Giann Carlos Spilere Nandi ; orientadora, Analúcia Schiaffino Morales, 2017.
123 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Campus Araranguá, Graduação em Engenharia de Computação, Araranguá, 2017.

Inclui referências.

1. Engenharia de Computação. 2. segurança da informação. 3. redes de sensores sem fio. 4. protocolos criptográficos. 5. verificação formal. I. Morales, Analúcia Schiaffino . II. Universidade Federal de Santa Catarina. Graduação em Engenharia de Computação. III. Título.

Giann Carlos Spilere Nandi

VERIFICAÇÃO FORMAL DE PROTOCOLOS DE SEGURANÇA VOLTADOS À REDES DE SENSORES SEM FIO

Este Trabalho de Conclusão de Curso foi julgado aprovado para a obtenção do Título de "Bacharel em Engenharia de Computação" e aprovado em sua forma final pela Universidade Federal de Santa Catarina.

Araranguá, 06 de dezembro de 2017.

Prof[a]. Eliane Pozzebon, Dra.
Coordenadora do Curso

Banca Examinadora:

Prof[a]. Analúcia Schiaffino Morales, Dra.
Universidade Federal de Santa Catarina

Prof. Gustavo Medeiros de Araújo, Dr.
Universidade Federal de Santa Catarina

Prof. Ricardo Alexandre Reinaldo de Moraes, Dr.
Universidade Federal de Santa Catarina

Aos meus pais...

AGRADECIMENTOS

Agradeço à Universidade Federal de Santa Catarina por todas as oportunidades à mim oferecidas durante a minha graduação. Agradeço também a minha orientadora, Profa. Dra. Analúcia Schiaffino Morales, por toda sua ajuda, o Prof. Dr. Gustavo Medeiros de Araujo pela cooperação durante o desenvolvimento do trabalho e ao Prof. Dr. Ricardo Alexandre Reinaldo de Moraes por ter aceito fazer parte da banca de defesa. Agradeço por fim meus pais, Clair Teresinha Spilere e Volnei Nandi por todo o apoio dado durante todas as fases da minha vida.

busquem conhecimento

et bilu

RESUMO

Cibersegurança não é um termo novo no âmbito computacional, entretanto, é o tipo de termo que consegue sempre tratar de temas atuais e relevantes. Uma das suas áreas de pesquisa trata de protocolos de segurança, também conhecidos como protocolos criptográficos. Estes protocolos descrevem sequências de ações a serem executadas por entidades a fim de alcançar certos objetivos de forma segura. Com a popularização do conceito de internet das coisas, novas necessidades surgiram no âmbito de segurança. Inserido no contexto de internet das coisas, a área de redes de sensores sem fio trouxe a necessidade de superar novos desafios em relação a segurança da informação. Dentre esses desafios adiciona-se a limitação de recursos computacionais, como baixa capacidade de processamento, baixa capacidade de armazenamento em memória, largura de banda reduzida e dispositivos alimentados por bateria. Diante dos novos desafios encontrados na redes de sensores, vários protocolos criptográficos foram propostos para atender a necessidade de garantir a segurança da informação, levando em consideração a limitação de recursos. Entretanto, devido ao projeto desses protocolos serem propenso a erros, brechas de segurança podem ser expostas e exploradas. Para identificar que propriedades de segurança são garantidas e a quais ataques pode estar exposto, protocolos podem ser submetidos a verificações de corretude, que fazem uso de métodos formais em suas análises. Este trabalho, visando enriquecer a escassa literatura sobre o assunto no Brasil, verifica formalmente três protocolos criptográficos para redes de sensores sem fio quanto as propriedades de segurança de sigilo e autenticidade.

Palavras-chave: segurança da informação. redes de sensores sem fio. protocolos criptográficos. verificação formal.

ABSTRACT

Cybersecurity is not a new term in the computational scope; however, it is the type of term that can always deal with current and relevant issues. One of its areas of research deals with security protocols, also known as cryptographic protocols. These protocols describe sequences of actions to be performed by entities to achieve specific goals securely. With the popularization of the internet concept of things, new needs have emerged within the scope of security. Inserted in the context of the internet of things, the area of wireless sensor networks has brought the need to overcome new challenges concerning information security. Among these problems is the limitation of computational resources, such as low processing capacity, low memory capacity, reduced bandwidth, and battery-powered devices. Given the new challenges encountered in sensor networks, several cryptographic protocols have been proposed to meet the need to ensure information security, taking into account resource limitations. However, because the design of these protocols is prone to errors, security breaches can be exposed and exploited. To identify which security properties are guaranteed and to which attacks may be detected, protocols can be submitted to verifications that use formal methods in their analyzes. This work, aiming to enrich the scarce literature on the subject in Brazil, formally verifies three cryptographic protocols for wireless sensor networks as the security properties of secrecy and authenticity.

Keywords: information security. wireless sensor networks. cryptographic protocols. formal verifications.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

LISTA DE SÍMBOLOS

SUMÁRIO

1 INTRODUÇÃO

Avanços tecnológicos relacionados a área de sensores, circuitos integrados e comunicação sem fio, como apresentado já por Loureiro (1997), proporcionaram um aumento significativo no número de projetos relacionados ao conceito de Internet das Coisas (*Internet of Things - IoT*). Este conceito é datado de 1999, quando Kevin Ashton foi pioneiro em descrever um sistema onde dados obtidos no mundo físico puderam ser enviados à internet graças ao uso de sensores (ROSE; ELDRIDGE; CHAPIN, 2015).

Aplicaçõe desta ideia desde então são as mais diversas. Um exemplo é apresentado em Khan (2017), onde diversos sensores são utilizados na obtenção de dados de pacientes em um hospital, sendo estas informações a base para a tomada de decisões da equipe médica.

Outro exemplo é apresentado em Chong e Ng (2016), onde dois estudantes da *Universiti Tunku Abdul Rahman* implementaram um sistema de gerenciamento de tráfego em uma área urbana. Através da análise de fluxo de veículos em cada pista e da logística desenvolvida a partir os dados coletados, o sistema foi capaz de diminuir em até 67% os engarrafamentos em cruzamentos.

Entretanto, dispositivos e sistemas *IoT* ainda enfrentam diversas barreiras de projeto e implementação. Dentre os desafios enfrentados, como explicado por Gazis et al. (2015), Lee e Lee (2015), Chase (2013), Ott (2016) e Kranenburg e Bassi (2012), pode-se citar a conectividade, escalabilidade, segurança e a privacidade e o consumo de energia, sendo a questão da segurança e privacidade das comunicações de dados o tema foco deste trabalho.

Não somente dispositivos *IoT*, mas diversas outras aplicações da atualidade fazem extensivo uso da troca de informações entre entidades distintas. Como explica Nogueira (2008), exemplos disso são diversos, passando por transações bancárias *on-line*, comércio virtual, correio eletrônico e comunicações militares. Logo, a manutenção de propriedades de segurança nestes tipos de aplicação são de vital importância para seus corretos funcionamentos.

Nota-se, entretanto, que existem múltiplas maneiras de se deturpar a segurança de sistemas de comunicação de dados. Nawir et al. (2016) as divide em dois grandes grupos. Ataques passivos, que ocorrem quando se visa obter informações da comunicação que ocorre entre dispositivos através de espionagem e monitoramento, e ataques ativos, que objetivam quebrar a segurança dos sistemas de forma que informa-

ções indesejadas sejam inseridas na comunicação ou fazendo com que meios de comunicação sejam prejudicados.

Um caso real de ataque a sistemas *IoT* é apresentado por Ronen et al. (2016), onde se retrata um caso de ataque bem sucedido contra um sistema de iluminação gerenciado através de comunicação sem fio. Através da análise do comportamento apresentado pelo hardware durante o processamento de dados criptográficos e da identificação de uma falha do protocolo de comunicação utilizado no aparelho, os autores do artigo em questão foram capazes de tomar controle do dispositivo e derivar os comandos essenciais para governar qualquer lâmpada do mesmo modelo.

Em poderio de centenas de lâmpadas infectadas, os autores eram capazes de interromper a iluminação de grande parte de uma cidade e ao mesmo tempo ter ao seu dispor um volume de dispositivos que poderiam ser explorados para, por exemplo, um massivo ataque de negação de serviço, também conhecido como *Distributed Denial of Service* (*DDoS*) . Ataques como este colocam vidas em risco e representam violações diretas à privacidade de indivíduos e organizações, visto que dispositivos como câmeras estão entre os mais vulneráveis a estes ataques, como explicam Cerrudo (2015), Korolov (2016) e Desai (2016).

Garantir que trocas de informações ocorram de forma segura independentemente da aplicação que as implementa ou dos dados por ela trocados é o que trata a área de segurança computacional. Com o objetivo de garantir a segurança da comunicação entre participantes, diversas medidas são tomadas visando garantir a manutenção de certas propriedades dos sistemas, dentre elas estão os protocolos de segurança.

1.1 DEFINIÇÃO DO PROBLEMA

Protocolos de segurança, segundo Carle (2003), são definidos como séries de passos a serem seguidos e mensagens a serem trocadas por duas ou mais entidades, objetivando alcançar uma meta de segurança desejada. Já Kremer (2006) apresenta protocolos criptográficos (outra maneira de identificar este tipo de protocolo) como pequenas aplicações distribuídas que objetivam garantir propriedades de segurança em ambientes hostis.

São diversas os protocolos que surgiram ao longo das últimas três décadas, cada um se propondo a garantir a manutenção de certas propriedades de segurança em diferentes tipos de redes e aplicações. Entretanto, como explica Nogueira (2008), o desenvolvimento de muitos

destes protocolos seguia/segue o seguinte ciclo de execução: proposta do protocolo, descoberta de algum tipo de ataque e reestruturação do protocolo até que mais alguma irregularidade fosse descoberta. Este ciclo faz com que protocolos estejam sempre expostos a novos ataques, visto que seus desenvolvedores não tem uma noção clara da totalidade do que seus protocolos são, ou não, capazes de assegurar.

Com o objetivo de obter garantias quanto ao funcionamento de protocolos e que propriedades de segurança eles são capazes ou não de manter, um campo de pesquisa chamado de verificação formal de protocolos criptográficos começou a ganhar espaço na comunidade de segurança computacional. Este tipo de técnica submete, de forma sistemática, protocolos criptográficos a rigorosas verificações lógicas e matemáticas. Estas verificações formais resultam em provas sobre o funcionamento dos protocolos, sendo estas provas utilizadas para determinar se o protocolo é ou não seguro.

Contudo, vale destacar que, como discutido em Santos (2012), o conceito de segurança não é algo único, mas sim a união de diversos fatores que devem ser levados em consideração ao se projetar um protocolo de segurança. Em outras palavras, os níveis de segurança exigidos variam de aplicação para aplicação, fazendo com que uma configuração seja ideal para certas situações e falha para outras. Em Nogueira (2008) o autor se aprofunda nesta discussão levantando que a segurança dos protocolos criptográficos é fortemente ligada às hipóteses levantadas durante seu desenvolvimento.

Como salientado em Santos (2012), o fluxo proposto em protocolos criptográficos independe do *hardware* onde são implementados. É esta sequência de eventos que é colocada à prova quando se deseja identificar possíveis falhas nos protocolos em seus âmbitos teóricos e formais. Em Nogueira (2008) e Santos (2012) discorre-se sobre a necessidade de se provar que as propriedades de segurança são realmente garantidas e como estas provas impactam na confiança que desenvolvedores de sistemas, aplicações e dispositivos tem com relação a segurança das informações por eles tratados.

Uma das áreas de pesquisa e aplicação que se beneficiou diretamente dos avanços com sensores e comunicação de dados é a de Redes de Sensores Sem Fio (RSSF) . Como descrito por Yick, Mukherjee e Ghosal (2008), RSSF são redes *wireless* constituídas de dispositivos autônomos (e de baixo poder de processamento e armazenamento) distribuídos espacialmente, os quais utilizam sensores para monitorar condições físicas ou ambientais e que são largamente utilizadas em aplicações IoT.

São diversas as áreas onde as RSSF podem ser empregadas,

podendo-se citar a agricultura, a pecuária, o monitoramento de redes inteligentes de energia e o tráfego urbano como exemplos. A fim de garantir a segurança das comunicações nos sistemas onde são adotados, os protocolos criptográficos (como o TinyPBC Oliveira et al. (2008)) apresentam maneiras de como configurar redes, realizar acordos de chaves e como proceder com relação a troca de informações entre participantes.

Um exemplo de verificação formal é apresentado por Chang e Shmatikov (2007) onde os autores submetem protocolos de pareamento *bluetooth* à verificações formais para analisar suas propriedades de autenticidade. Os autores ainda fazem uso da mesma ferramenta de verificações formais a ser utilizada neste trabalho.

Já Bursuc (2016) verifica formalmente um protocolo voltado à sistemas de gerenciamento de conferências utilizado em aplicações que envolvem computação nas nuvens. Aqui os autores também se utilizaram da ferramenta ProVerif para executar suas validações formais.

Outro exemplo de verificação formal pode ser encontrado no texto de Shaikh, And e Devane (2010), onde os autores utilizam outra ferramenta de verificação formal, chamada de AVISPA, para analisar a garantia das propriedades de segurança de um protocolo de pagamentos digital.

Direcionando então o foco das verificações formais para área de redes de sensores sem fio, tem-se como exemplo o trabalho apresentado por Kasraoui, Cabani e Chafouk (2014), onde os autores propões um protocolo para este tipo de rede e realiza a verificação formal dele através também da utilização do software AVISPA.

Como último exemplo, o trabalho de Islam (2015) se propõe a realizar a verificação formal de um protocolo que já havia sido publicado, porém não havia passado por verificações formais, sendo o mesmo procedimento adotado pelo autor do presente documento.

Dentre diversos protocolos que existem atualmente, foram escolhidos três para serem formalmente verificados. Estes protocolos são descritos por Oliveira et al. (2008), Herrera e Hu (2012) e Saqib (2016), visto que, do que é de conhecimento do autor deste trabalho, não existem verificações formais realizadas até o momento para este protocolos.

1.2 JUSTIFICATIVA

Como apresentado anteriormente, do que é de conhecimento do autor deste trabalho, não existem verificações formais dos três protocolos escolhidos. Logo, aqueles que cogitem os utilizar em suas aplicações

não possuem provas quanto a corretude deste protocolos. Além disso, protocolos com resultados positivos em verificações formais geram interesse comercial em diversas áreas de aplicação, visto que independem do hardware que o aplica.

Assim, como comenta Piva (2009), a utilização de métodos de validação formal, fornece aos desenvolvedores de futuras aplicações *IoT*, provas quanto as garantias e falhas que protocolos criptográficos apresentam.

Por fim, este trabalho visa também contribuir com a, escassa Santos (2012), literatura de validação formal de protocolos criptográficos voltados a RSSF.

1.3 HIPÓTESE

Utilizando métodos formais para verificar os protocolos (OLIVEIRA et al., 2008), (HERRERA; HU, 2012) e (SAQIB, 2016) é possível provar a garantia, ou não, de suas propriedades de segurança de sigilo e autenticidade.

1.4 OBJETIVOS

Afim de definir o escopo do projeto, é apresentado nesta seção o objetivo geral juntamente com os objetivos específicos.

1.4.1 Objetivo Geral

Verificar se as propriedades de sigilo e autenticidade dos protocolos criptográficos propostos por Oliveira et al. (2008), Herrera e Hu (2012) e Saqib (2016) são, ou não, garantidas.

1.4.2 Objetivos Específicos

- Criar modelos capazes de representar os processos e as características dos protocolos criptográficos propostos por Oliveira et al. (2008), Herrera e Hu (2012) e Saqib (2016);

- Validar formalmente os protocolos criptográficos propostos por Oliveira et al. (2008), Herrera e Hu (2012) e Saqib (2016);

- Identificar possíveis ataques aos quais os protocolos propostos possam estar expostos;
- Identificar quais as propriedades que cada um dos protocolos é capaz de garantir.

1.5 MÉTODOS DE PESQUISA

Para atingir os objetivos apresentados na seção 1.4, foi definida uma estrategia composta pelos seguintes passos

1. Criação de modelos capazes de representar detalhadamente o funcionamento dos protocolos a serem analisados;
2. Transcrição dos modelos criados para a linguagem *Applied Pi Calculus*;
3. Utilizar a ferramenta de validação formal ProVerif para avaliar os modelos transcritos;
4. Analisar os resultados obtidos das validações e verificar quais as propriedades de segurança são, ou não, garantidas.

1.6 ORGANIZAÇÃO DO DOCUMENTO

O capítulo **2** tem como objetivo apresentar definições sobre termos da área de segurança de comunicação de dados, além de apresentar propriedades e mecanismos de segurança. O capítulo **3**, por sua vez, aprofunda o tema de segurança computacional direcionando o foco para as aplicações que envolvem a utilização de redes de sensores sem fio. No capítulo **4** se trata de definições sobre a verificação formal de protocolos criptográficos. O capítulo **5** objetiva explicar ao leitor o funcionamento dos três protocolos criptográficos a serem formalmente verificados. No capítulo **6** são apresentados os resultados obtidos através das verificações formais dos protocolos juntamente com uma apresentação da ferramenta utilizada para realizar estas verificações. Por fim, no capítulo **7** são abordadas as considerações finais sobre o trabalho e propostas de trabalhos futuros.

2 SEGURANÇA EM SISTEMAS DE COMUNICAÇÃO DE DADOS

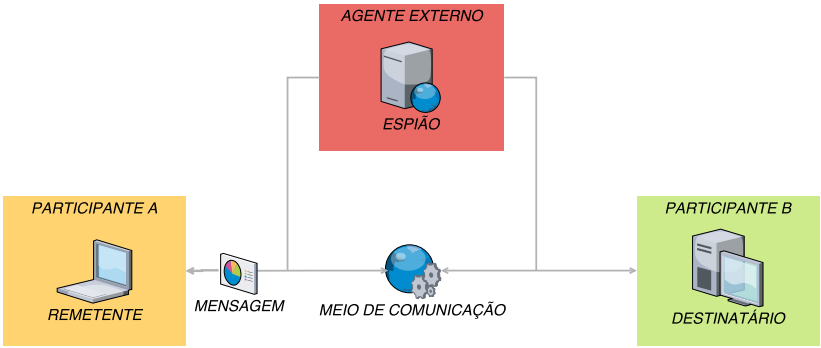
O presente capítulo apresenta os principais conceitos de segurança de sistemas de comunicação de dados tendo em vista o escopo abordado por este trabalho. Destacam-se as principais técnicas e ferramentas empregadas em segurança de sistemas que são essenciais para a compreensão do que se é apresentado ao longo do documento.

2.1 MODELO DE COMUNICAÇÃO

O primeiro aspecto a ser abordado neste capítulo trata do modelo adotado nas comunicações de dados dos protocolos a serem estudados. Vale ressaltar que o modelo tem como objetivo apresentar os aspectos da comunicação em nível de descrição de protocolos de comunicação de dados.

Stallings (2010) descreve um modelo onde existem cinco principais elementos: um remetente, uma mensagem, um canal de comunicação, um destinatário e um possível espião.

Figura 1 – Modelo de Comunicação



Fonte: do autor

O remetente é aquele que deseja enviar algum tipo de informação, sendo definida como a mensagem neste modelo, a um outro participante

da rede. Tal mensagem é enviada através de um canal de comunicação onde, no caso das redes *wireless*, é aberto e qualquer um pode enviar e receber mensagens desde que corretamente configurado. O destinatário é aquele ao qual a mensagem inicial era destinada. Por fim, o espião é um agente externo à comunicação que objetiva obter informações e/ou obstruir o processo de comunicação entre os participantes da rede. A figura 1 é utilizada para ilustrar este modelo.

2.2 SEGURANÇA COMPUTACIONAL

A segurança computacional engloba todos os aspectos relacionados a segurança de redes, sendo definida por Guttman e Roback (1995) como a proteção oferecida a um sistema de informações automatizado com o objetivo de garantir as propriedades de **integridade**, **disponibilidade** e **confidencialidade** dos recursos do sistema de informação.

Essa definição traz três pontos chave para o tema de segurança, como explicado por Stallings (2010) e Evans, Bond e Bement (2004):

2.2.1 Confidencialidade

Representa a preservação de restrições autorizadas ao acesso e descoberta de informações. Uma perda de confidencialidade é resultado da descoberta de informações de forma não autorizada. Este conceito é subdividido em dois grupos:

- Confidencialidade de Dados: assegura que informações privadas e/ou confidenciais não estejam disponíveis ou sejam descobertas por indivíduos não autorizados;
- Privacidade: assegura que indivíduos tenham controle com relação a quem tem ou não acesso as suas informações privadas.

2.2.2 Integridade

Representa a proteção contra a destruição ou modificação indevida de informações. Este conceito é subdividido em dois grupos:

- Integridade de Dados: assegura que informações e programas sejam modificados apenas de maneiras especificadas e autorizadas;
- Integridade dos Sistemas: assegura que os sistemas executem suas

funções previstas de forma intacta, sem manipulação não autorizada dos sistemas.

2.2.3 Disponibilidade

Assegura que o sistema é capaz de, em tempo aceitável para a aplicação, permitir o acesso a dados e o uso dos mesmos.

Além dos três grandes grupos acima, Stallings (2010) adiciona a esta lista mais dois pontos chaves de segurança:

2.2.4 Autenticidade

A propriedade responsável por verificar se um indivíduo é quem diz ser, e se toda informação recebida vem realmente do remetente confiável que do qual diz vir.

2.2.5 Responsabilidade

A propriedade que garante que existam registros de atividades dentro do sistema, fazendo com que haja a possibilidade de se identificar a origem de problemas que podem vir a ocorrer.

2.2.6 Arquitetura de Segurança para Interligação de Sistemas Abertos

Aprovado em 1991, o texto conhecido como Recomendações ITU-T X.800 ITU (1991) é um documento referência na área de segurança em comunicação de dados que traz uma descrição geral de serviços de segurança e os mecanismos de segurança a eles relacionados. Essas definições continuam sendo utilizados mais de duas décadas depois de sua publicação e servem como base para muitos dos termos a serem apresentados a seguir. Segundo o documento, sistemas de segurança são composto de:

- Ataques de Segurança: qualquer ação que comprometa a segurança das informações de organizações;
- Mecanismos de Segurança: processos que se dedicam a detectar,

prevenir ou se recuperar de ataques de segurança;

- Serviços de Segurança: serviços de processamento ou comunicação que aumentam o nível de segurança das informações processadas e transmitidas por organizações. Serviços de segurança fazem uso de mecanismos de segurança para proteger dados contra ataques de segurança.

2.3 TIPOS DE ATAQUE

De forma geral, como apresentado por Stallings (2010) e Nawir et al. (2016), ataques de segurança podem ser classificados em dois grandes grupos, ataques passivos e ataques ativos. O primeiro deles é aquele em que se visa aprender algo sobre ou utilizar de alguma forma as informações transmitidas pelos sistemas, entretanto, não existe nenhum tipo de interferência que venha a afetar o processo de comunicação.

Ataques passivos se baseiam, basicamente, em processos de espionagem e monitoramento da troca de informações. Com isso, o agente mal intencionado busca, de alguma forma, descobrir as informações que estão sendo transmitidas. Estes ataques se dividem em dois tipos, como apresentado por Stallings (2010):

- Liberação de Conteúdo: ocorre quando entidades não autorizadas conseguem obter informações sendo trocadas entre dois participantes de forma aberta. Em outras palavras, é quando um agente mal intencionado consegue obter os dados que estão sendo transmitidos devido ao fato de não se tenta omitir as informações de maneira alguma;

- Análise do Tráfego: este tipo de ataque ocorre quando, de alguma forma, os participantes honestos conseguem disfarçar a mensagem original e existem agentes externos tentando identificar padrões do disfarce empregado aos dados.

Stallings (2010) afirma que este tipo de ataque dificilmente é identificado, visto que em nenhum momento as informações trocadas são modificadas, tornando quase impossível identificar um agente externo nas comunicações. Nota-se, entretanto, que impedir que ataques passivos sejam bem sucedidos é possível graças a técnicas de criptografia (tópico a ser discutido mais adiante neste trabalho).

O segundo grupo trata de ataques ativos, onde agentes mal intencionados interferem de alguma forma na troca de informações entre

os participantes de uma comunicação. Estes ataques podem ocorrer das seguintes maneiras:

- *Replay*: ocorre quando mensagens são capturadas e retransmitidas a fim de afetar de alguma forma o sistema;
- Modificação de Mensagens: aqui a mensagem originalmente transmitida é modificada de alguma maneira, seja em seu conteúdo, atraso em sua chegada ou reordenando sua entrega, por exemplo;
- Negação de Serviço: este tipo de ataque tem como objetivo obstruir e impedir a livre utilização dos recursos de comunicação da rede. Pode-se ou não se escolher alvos em específico para executar a perturbação;
- Disfarce: ocorre quando um participante externo não autorizado assume a identidade de participantes honestos dentro dos processos.

2.4 SERVIÇOS DE SEGURANÇA

Serviços de segurança são definidos em ITU (1991) como serviços providos à camadas de protocolos e que visam assegurar um adequado nível de segurança a sistemas ou transferência de dados. Uma outra maneira de definir este tipo de serviço é apresentada por Shirey (2000), onde se afirma que serviços de segurança são processos implementados em sistemas para proteger certos recursos, criando-se políticas de segurança através de mecanismos de segurança (tema a ser explicado na próxima seção).

Guttman e Roback (1995) e ITU (1991) dividem os serviços em cinco categorias, sendo elas explicadas por Stallings (2010) como:

2.4.1 Autorização

Objetiva assegurar que comunicações são autênticas. No caso da troca de mensagens entre dois participantes, a autenticação ocorre quando se assegura que a mensagem entregue ao destinatário partiu do remetente que a diz ter mandado. Já no caso de interações contínuas, é necessário que, ao se iniciar a comunicação, verifique-se a autenticidade de ambos os participantes. É exigido também que o serviço garanta que comunicações não sejam afetada por agentes externos que se disfarcem de entidades honestas do processo.

2.4.2 Controle de Acesso

Trata da garantia de que apenas as entidades autenticadas e autorizadas tenham acesso a certas funcionalidades do sistema, controlando o acesso dos usuários.

2.4.3 Confidencialidade

Abrange a questão da proteção dos dados contra ataques passivos, fazendo com que agentes mal intencionados não sejam capazes de identificar padrões através da análise da comunicação. Dentre os possíveis padrões que podem ser alvo de análise se pode citar o destino, a fonte e até mesmo o tamanho de uma mensagem;

2.4.4 Integridade dos Dados

Visa garantir que as mensagens recebidas estejam intactas, ou seja, não sofram modificações durante transferências. Dentre as possíveis alterações de mensagens se pode citar a duplicação, inserção de dados, modificação de conteúdo, reordenação e repetição. A destruição de mensagens também é abordada por este tipo de serviço.

2.4.5 Não Repúdio

Trata de garantir que nem o remetente possa negar ter enviado uma mensagem, nem o destinatário possa negar a ter recebido. Dessa forma participantes, ao receberem informações, podem provar que remetentes de fato as enviaram e, de forma similar, remetentes podem confirmar que destinatários receberam mensagens pelos remetentes enviadas.

2.4.6 Disponibilidade

Segundo Shirey (2000), disponibilidade está relacionada ao desimpedimento do uso de sistemas (ou recursos deles) sob a demanda de entidades autorizadas e verificadas, obedecendo as especificações de desempenho pelo sistema estabelecidas. Em geral, este serviço tem como

objetivo assegurar que ataques de negação de serviço não tenham exito em seus processos.

2.5 MECANISMOS DE SEGURANÇA

Como visto na seção anterior, mecanismos de segurança são ferramentas utilizadas pelos serviços de segurança para garantir que ataques de segurança não sejam bem sucedidos. Baseado no texto de Guttman e Roback (1995), Stallings (2010) alguns exemplos de tais mecanismos são apresentados:

2.5.1 Cifragem

Para Tripathi e Agrawal (2014), a cifragem de informações é definida como a arte e a ciência de proteger informações contra participantes não desejados, transformando conjuntos de dados em algo indistinguível quando comparado com a informação original. Este tipo de abordagem é de essencial importância quando se deseja realizar a troca de informações de forma segura em canais de comunicação inseguros.

Para que se possa introduzir o tema de cifragem, alguns termos chave são apresentados então baseando-se no texto de Stallings (2010):

- Texto Plano: é a mensagem original a ser utilizada como valor de entrada para o algoritmo de cifragem;
- Texto Criptografado: é o resultado das modificações aplicadas ao texto plano por um algoritmo de criptografia, o tornando ilegível;
- Chave: valor utilizada no processo de cifragem para transformar textos planos em textos criptografados, ou no processo de descriptografia, que retorna textos cifrados em textos planos;
- Cifragem ou algoritmo de criptografia: é o responsável por transformar textos planos em textos criptografados;
- Descriptografia: processo reverso ao processo de cifragem, transformando textos criptografados em textos planos. Vale ressaltar que aqui se tem ciência de como ocorreu o processo de cifragem, diferentemente da criptoanálise;

- Criptoanálise: são técnicas que objetivam decifrar o que está contido em textos criptografados através da análise das mensagens. No processo de criptoanálise, o conhecimento parcial sobre a criptografia, em certos casos, já é suficiente para que criptoanalistas possam inferir o conteúdo da mensagens cifradas.

Algoritmos de criptografia trabalham com basicamente duas operações para cifrar textos planos, como explicam V.Hemamalini et al. (2016), são elas:

- Técnicas de Substituição: textos planos são substituídos por textos cifrados seguindo determinados padrões e guiado por chaves a serem utilizadas. Exemplos de técnicas como esta são as Cifras de César e as Cifras Monoalfabéticas;

- Técnica de Transposição: trabalha com a permutação de caracteres de textos planos. Como exemplos deste tipo de técnica, pode-se citar a transposição Dupla e a transposição de Myszkowski.

Por fim, classificam-se os tipos de algorítimos criptográficos conforme as chaves a serem utilizadas.

2.5.1.1 Criptografia Simétrica

Também conhecida como criptografia convencional, ou ainda, criptografia de chave única, como apresentado por Stallings (2010), foi o primeiro método a ser utilizado no meio computacional, sendo também o único até a década de 70, quando foram apresentados ao mundo, através do texto de Diffie e Hellman (1976), os protocolos envolvendo criptografia de chave pública.

O processo de comunicação neste tipo de criptografia ocorre da seguinte forma: um usuário qualquer **A** deseja enviar uma mensagem criptografada para um usuário **B**. Para isso, cifra-se o texto plano com uma chave que tanto **A** como **B** possuem e que, em teoria, somente eles conhecem. Ao receber a mensagem cifrada, **B** utiliza a chave simétrica para aplicar a função de descriptografar a mensagem.

Na criptografia simétrica, a chave utilizada tanto no processo de encriptação do texto plano em texto criptográfico, como também na descriptografia do texto cifrado em texto plano é a mesma. Por consequência, ambos participantes da comunicação devem conhecer a priori seu valor.

Essa exigência faz com que exista a necessidade da existência de canais seguros de comunicação onde se consiga realizar a distribuição das chaves a serem utilizadas (V.HEMAMALINI et al., 2016). Este procedimento é inviável em aplicações onde os participantes nunca tiveram um contato prévio ou, não possuam intermediadores em comum. Além disso, como apresentado por Stallings (2010), é necessário também a existência de entidades que sejam responsável por administrar e gerenciar as chaves utilizadas pelos participantes.

Apesar das desvantagens acima mencionadas, o processamento de criptografia de chave simétrica tem ótimo desempenho computacional e, até o momento, continua sendo utilizado em diversas aplicações.

2.5.1.2 Criptografia Assimétrica

Diferentemente da criptografia simétrica, a cifragem do texto plano e a descriptografia do texto criptografado se dão através de chaves distintas. Este tipo de criptografia é também conhecida como criptografia de chave pública, ou criptografia de duas chaves. Sua primeira aparição na literatura ocorreu através do trabalho de Diffie e Hellman (1976), onde os autores apresentam uma proposta onde dois usuário distintos e desconhecidos conseguem se comunicar de forma segura através de uma combinação de chaves onde certas informações são públicas, ou seja, acessíveis por qualquer um, enquanto outras informações são privadas e apenas certos participantes as conhecem.

O trabalho deles impactou de tal forma a área de segurança computacional, que se dividiu a história da criptografia em duas eras, a clássica e a moderna, sendo a publicação de Diffie e Hellman (1976) o marco da divisão. Denominou-se a segunda era como moderna devido ao fato de, pela primeira vez, a teoria de números ter sido utilizada como base para a criptografia.

O sucesso do trabalho de Diffie e Hellman é, em grande parte, atribuído a introdução do conceito de funções unidirecionais na criptografia. Este conceito traz que funções sejam facilmente calculáveis em uma direção, porém computacionalmente impossíveis de se descobrir suas inversas sem o conhecimento da função originalmente utilizada. Este tipo de função possui um termo comumente utilizado na criptografia como *trapdoor function*. Simmons (1979) explica que este comportamento se dá pela facilidade em se obter resultados de operações de multiplicação e adição utilizando números primos de grande magnitude, e a dificuldade em se encontrar a fatoração desses números, o que

torna a operação não factível computacionalmente.

Na criptografia assimétrica os participantes da comunicação possuem pares de chaves, uma chamada de chave pública e outra chamada de chave privada. As chaves públicas são exatamente o que o nome sugere, elas estão disponíveis para qualquer agente, seja ele honesto na rede ou não, que deseje as utilizar para cifrar informações. Já as chaves privadas são mantidas em segredo por cada um dos usuários. Em geral, é desejável que os pares de chaves sejam únicos no grupo de participantes que se está trabalhando.

Criptografias de chave pública funcionam de tal maneira que, para se descriptografar uma informação, é necessária a utilização de uma combinação de chaves públicas e privadas. A combinação mais comum se dá quando um participante **A** envia uma mensagem para um outro participante **B** a encriptando utilizando a chave pública de **B**. **A**, sabendo que apenas **B** possui a chave necessária descriptografar a mensagem, sabe, portanto, que o conteúdo então não será visto por nenhum participante externo.

Ao contrário do que ocorre nos processos de criptografia de chave única, as criptografias de chave pública não necessitam de um canal seguro onde as entidades realizam trocas de chaves a serem utilizadas na cifragem. Aqui, o sucesso ou fracasso da segurança fornecida pelo protocolo depende da confidencialidade das chaves privadas de cada um dos usuários. Dentre os diversos algoritmos de criptografia de chave pública, são detalhados os mais relevantes para este trabalho:

2.5.1.2.1 Algorítimo RSA

Desde a apresentação do trabalho pioneiro de (DIFFIE; HELLMAN, 1976), marco na introdução do que se chama atualmente de criptografia moderna, diversos algoritmos foram propostos ao longo dos anos. Dentre eles, um se destaca até hoje por ser o mais utilizado desde então, o esquema RSA , nome vindo das iniciais do sobrenome de cada autor do trabalho (RIVEST; SHAMIR; ADLEMAN, 1978).

Neste esquema de criptografia, calcula-se um valor máximo a ser utilizado no processo criptográfico, resultado da multiplicação de dois números primos aleatórios. As chaves públicas e privadas são dois números especialmente escolhidos e que satisfazem duas condições: são maiores do que zero e menores do que o valor máximo.

O algoritmo utiliza cifragem de blocos, onde o texto plano e o texto criptografado são inteiros que variam entre 0 e n-1 para um n

qualquer. Este valor de n é um dos fatores que determina o nível de segurança empregado ao algoritmo. Em geral, como comentado em (STALLINGS, 2010), o valor de n é maior ou igual a 1024bits. Valores menores do que este costumam não fornecer um nível de segurança confiável.

O funcionamento do algoritmo é composto pelos seguintes procedimentos. Primeiramente, são escolhidos dois números primos aleatoriamente, chamando-os de p e q. O resultado do produto é valor máximo a ser utilizado durante o algoritmo, chamado de $n = pq$. Calcula-se então z,

$$z = (p - 1)(q - 1)$$

Em posse de n e de z calcula-se e, valor utilizado para encriptar a mensagem. Entretanto, e deve satisfazer duas condições:

1. não possua fatores comuns com z

2. e deve ser menor que n

Para decifrar a mensagem, é necessário encontrar a chave correspondente a chave privada e, que neste caso tem o nome de d. O valor de d deve ser resultado da seguinte equação:

$$ed(mod\, z) = 1$$

Com isso, um participante pode encriptar uma mensagem utilizando o par de chaves (e,n) e um outro participante pode descriptografar a mensagem utilizando o par de chaves (d,n).

Para ilustrar o funcionamento do algoritmo é apresentado aqui um exemplo retirado de uma aula ministrada pelo professor Woo (2014). Primeiramente, escolhem-se dois números primos quais queres, sendo neste caso $p = 2$ e q=7. É importante destacar que tais valores são baixos somente neste exemplo, visto que valores maiores (que normalmente são utilizados neste algoritmo) dificultariam o entendimento dos passos.

O produto de p e q portanto é n, dado por:

$$n = 2 * 7$$

Em posse de n, calcula-se $\phi(n)$, dado por:

$$\phi(n) = (2 - 1)(7 - 1) = 6$$

Para a escolha de e, é necessário que se verifique que valores satisfazem as condições previamente mencionadas. A primeira delas é que o número esteja, neste caso, entre 1 e 5. A segunda é que o valor não seja fator nem de n e nem de $\phi(n)$, sendo 5 o único valor que obedece as condições propostas.

Com isso, encontrou-se o primeiro par de chaves a ser utilizado, sendo e a chave privada e n a chave pública para o processo de criptografia. É necessário agora que se escolha o par de chaves a ser utilizado na descriptografia. Para isso, é necessário escolher o valor de d:

$$5d(mod\phi(6)) = 1$$

Realizando o cálculo acima, percebe-se que existem diversos possíveis valores para d, sendo o valor escolhido para este exemplo o número 11. Dessa forma, tem-se agora o par de chave para a descriptografia da mensagem (11,14).

Para mostrar a criptografia e a descriptografia do algoritmo RSA, os valores acima calculados são então utilizados no exemplo abaixo, onde um participante **A** deseja enviar o algarismo 2 a um participante **B** através de um canal não seguro. **A** então calcula:

$$2^5(mod\ 14) = 4$$

O valor criptografado de 2 então é 4, sendo este o número enviado ao participante **B**. **B**, por sua vez, ao receber 4, e em posse do par de chaves que descriptografa o texto recebido, calcula:

$$4^{11}(mod\ 14) = 2$$

Obtendo, assim, o texto plano original.

2.5.1.2.2 Criptografia de Curva Elíptica

Em geral, a utilização de RSA como padrão de criptografia de chave pública é muito mais vasta do que a de outras abordagens da atualidade. Porém, como mencionando anteriormente, a segurança de tal esquema é dependente majoritariamente do tamanho do valor de n escolhido, fazendo com que a escolha de altos valores para a criação das chaves seja um empecilho para aplicações onde não se possui um poder de processamento compatível com tais grandezas.

Além disso, Sullivan (2013) explica que já existem algoritmos que tentam encontrar os fatores utilizados pelo algoritmo RSA. Tais

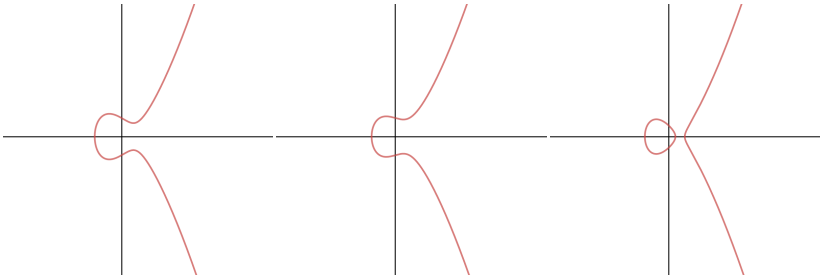
algoritmos trabalham de tal forma que a dificuldade em se encontrar os fatores diminui conforme o tamanho das chaves utilizadas aumenta. Dessa forma, a diferença existente entre o cálculo dos produtos e das fatorações tende a diminuir, fugindo do comportamento ideal das funções de mão única, que descreve que a dificuldade em se calcular os produtos e os fatores deve crescer proporcionalmente.

Com a proposta de diminuir o tamanho das chaves utilizadas mantendo um nível de segurança equivalente ao proposto pelo esquema RSA, surge a criptografia de curva elíptica (*Eliptic Curve Cryptography - ECC*) em 1985. Este tipo de criptografia faz uso de um tipo de aritmética própria, chamada de aritmética de curva elíptica. Entretanto, apesar de não ser uma área de estudo nova, apenas em meados de 2010 que se começou a gerar material com o objetivo de provar a resistência e as deficiências de tal técnica.

Curvas elípticas são conjuntos de pontos que satisfazem uma equação matemática específica. Estas equações seguem o seguinte padrão:

$$y^2 = x^3 + ax + b$$

Figura 2 – Exemplos de curvas elípticas.



Fonte: do autor.

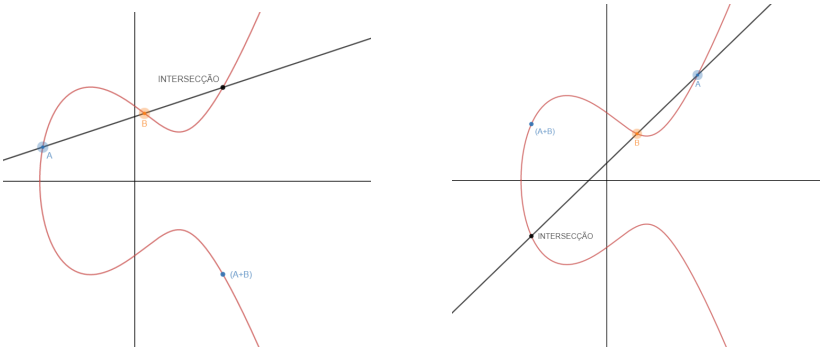
Dependendo dos valores adotados para as variáveis apresentadas na equação acima, a curva elíptica pode assumir infinitas formas. Na Figura 2 são apresentados três possíveis gráficos resultantes dessas variações.

Curvas elípticas apresentam algumas características bem peculiares. Observando a imagem acima percebe-se que existe uma simetria com relação ao eixo das abscissas. Além disso, ao se traçar uma linha

reta entre dois pontos quais queres da curva, essa reta interceptará a curva em no máximo 3 pontos no total. Essas características serviram como base para o que se conhece como aritmética de curvas elípticas. Este tipo de aritmética segue suas próprias definições, se distanciando quase que completamente da aritmética tradicionalmente conhecida. Aqueles que tenham interesse em conhecer um pouco mais sobre a aritmética de curvas elípticas podem consultar o texto introdutório escrito por Corbellini (2015).

Sullivan (2013) faz uma comparação muito pertinente e esclarecedora sobre como funciona este tipo de criptografia. Em seu texto, ele compara as operações realizadas na curva com um jogo de bilhares. Neste jogo, movimenta-se uma bola de um ponto A em direção a um ponto B, o ponto onde a bola intercepta a curva (não sendo nenhum dos pontos iniciais) fará com quem que a bola salte em em linha reta para baixo ou para cima, dependendo se ela está acima ou abaixo do eixo das abscissas, respectivamente. As figuras a seguir ilustram este comportamento.

Figura 3 – Adição em Aritmética de Curvas Elípticas



Fonte: do autor.

Este tipo de operação pode ser repetida n vezes, sendo que, mesmo se conhecendo o valor inicial e o final, é computacionalmente impossível de descobrir o número operações transcorridas durante o processo. Seguindo a mesma linha de pensamento do jogo de bilhar, é como se uma pessoa ordinária estivesse jogando por um tempo aleatório qualquer até que um indivíduo entre na sala e veja a disposição das bolas na mesa. Mesmo o indivíduo conhecendo as regras do jogo e

sabendo a posição atual das bolas e onde elas começaram, é impossível dizer quantas vezes as bolas foram atingidas para alcançarem seu estado final sem que o observante tenha passado pelo exato mesmo processo que o jogador inicial passou. Ou seja, é um processo fácil de fazer, porém difícil de desfazer, comportamento ideal esperado de uma função de mão única.

Fazendo a relação com o que foi visto com relação a chaves públicas e privadas, para realizar a criptografia utilizando curvas elípticas, escolhe-se inicialmente uma equação da curva. É importante destacar aqui que, apesar de existirem incontáveis formatos para as equações, existe um número limitado delas que são consideradas ideais para serem utilizadas neste tipo de operação. Desta curva então escolhe-se um ponto inicial a ser utilizado por todos participantes do protocolo. Como chave privada, cada participante recebe um valor n que determinará o número de vezes que as operações aritméticas ocorrerão. Por sua vez, o valor da chave pública é o ponto inicial passando pela operação n vezes.

Mesmo após mais de três décadas de pesquisa ainda não existe nenhuma proposta de como fatorar o valor n das curvas elípticas. Para se ter uma ideia da diferença no tamanho das chaves necessárias quando se utiliza curvas elípticas quando comparadas com RSA, uma chave de 228bits em ECC é equivalente a uma chave de 2380bits em RSA.

2.5.2 *Hash*

Funções *Hash*, conhecidas também como resumo criptográfico, aplicam o conceito de *trapdoor function*, porém sem a criação de chaves inversas para retornar ao texto plano utilizado, como explica O'Shea (2010). Aceita-se como entrada da função textos planos de tamanho variável e são produzidos textos criptografados de tamanho fixo e que garantem duas propriedades:

1. ser computacionalmente inviável de se encontrar os dados iniciais utilizado na criptografia;

2. impossibilidade de que se forneçam dois textos distintos que sejam mapeados para o mesmo valor final.

Funções *hash* podem ser utilizadas de diferentes formas, como explicam Stallings (2010), Grah (2008). Os protocolos presentes neste documento utilizam de duas formas principais:

- 1) verificar a integridade de mensagens. Um exemplo prático disso pode ser verificado ao se supor duas entidades que desejam trocar mensagens e necessitam de uma garantia de que as informações não foram alteradas durante o processo de envio. Para realizar esta verificação, o remetente envia um valor *hash* do texto plano concatenado ao próprio texto plano criptografado com uma chave qualquer. Ao receber a mensagem criptografa e, assumindo que o destinatário possui conhecimento necessário para descriptografar a mensagem, o recebedor, que também conhece a função *hash* utilizada, utiliza o texto plano descriptografado em sua função *hash* e compara o valor obtido e o valor recebido. Caso os valores sejam iguais, tem-se a garantia de que mensagem não foi alterada durante o processo.

- 2) aplicar de forma prática as funções de via única. Neste caso, se deseja cifrar informações confidenciais através de funções *hash*, criando assim dados que podem ser utilizados em diversas aplicações. As informações já transformadas podem ser divulgadas sem que o valor original seja ser descoberto, visto que funções de via única, como visto anteriormente, fazem com que informações utilizadas como entrada não sejam deriváveis do valor de saída.

2.5.3 Assinatura Digital

Como mencionado anteriormente, o mecanismo de assinatura digital é uma derivação do que se conhece dos procedimentos de criptografia de chave pública. Stallings (2010) considera, na verdade, que o mecanismo de assinatura digital foi o aspecto mais importante proveniente da criptografia de chave pública.

O processo de assinatura difere do que foi visto até agora com os algoritmos RSA e *ECC* onde se encriptava as mensagens com a chave pública do indivíduo que deveria receber a mensagem assumindo que somente ele a conseguiria descriptografar. Aqui, o processo é quase que invertido, a chave utilizada na criptografia é a privada do participante que deseja **enviar** a mensagem, sendo que qualquer um que possua sua chave pública pode ter acesso a mensagem.

Diferentemente do que acontece nos algoritmos criptográficos de chave pública, aqui o objetivo não é fazer com que apenas um indivíduo (ou um grupo de indivíduos) tenha acesso a informação a ser transmitida, mas sim, fazer com que todos aqueles possuam a chave pública do remetente tenham certeza sobre a identidade de quem enviou a mensa-

gem.

Assim como acontece nas assinaturas manualmente feitas em papéis, a assinatura digital tem como objetivo provar que quem assinou uma determinada mensagem é quem realmente se diz ser. Este tipo de verificação é feita partindo da ideia de que cada indivíduo na rede possui sua própria chave privada, e que, caso alguma mensagem tenha sido cifrada com tal chave, somente ele poderia ter realizado esta operação.

Por fazer com que a mensagem assinada seja acessível por qualquer indivíduo na rede, não é comum enviar mensagens em forma de texto plano assinadas digitalmente, ao invés, utiliza-se o resultado da aplicação de funções de resumo criptográfico, vistas na subseção acima, e se assina tais valores utilizando a técnica de assinatura digital.

3 SEGURANÇA EM REDES DE SENSORES SEM FIO

A segurança das redes tradicionais difere parcialmente quando comparadas as redes de sensores sem fio, visto que suas arquiteturas divergem em pontos intrínsecos de suas aplicações. Em suma, este capítulo apresenta algumas das diferenças que redes de sensores sem fio apresentam com relação as redes tradicionais. As propriedades e os ataques aqui apresentados servem de base para que o leitor consiga associar e perceber a utilidade das verificações formais a serem apresentadas mais adiante no documento.

3.1 REDES DE SENSORES SEM FIO

Os progressos relacionados às pesquisas nas áreas de circuitos integrados e comunicações de dados sem fio fizeram com que, em meados de 2010, o tema de redes de sensores sem fio (RSSF) ganhasse reconhecimento e destaque mundial (YICK; MUKHERJEE; GHOSAL, 2008). Este tipo de rede se diferencia das redes de computadores tradicionalmente conhecidas em diversos aspectos.

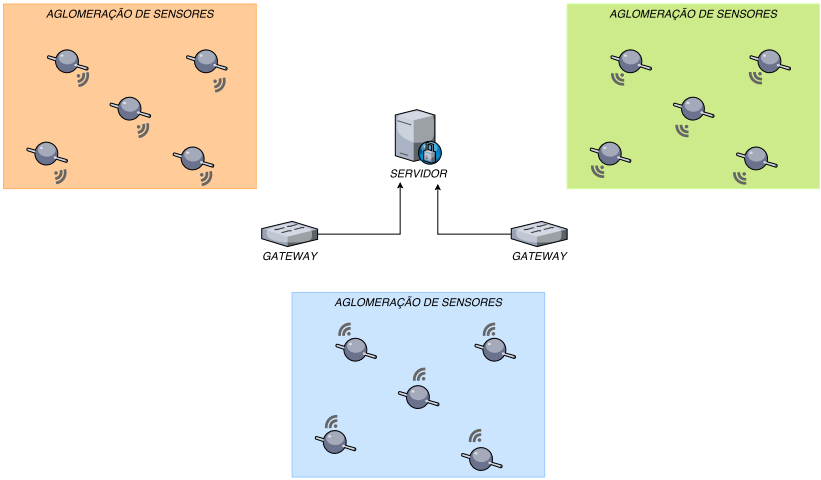
As RSSF são compostas, como descrito por Akyildiz et al. (2002) e Kadri et al. (2012), por pequenos aparelhos dotados de um ou mais sensores, um dispositivo de rádio para viabilização das comunicações sem fio, uma fonte de energia e de uma unidade de processamento própria, capaz de tratar as informações obtidas através das unidades sensoriais e de as enviar de forma tratada na rede.

Estas redes em geral não possuem uma estrutura sofisticada de organização (YICK; MUKHERJEE; GHOSAL, 2008). As redes chamadas de **não-estruturadas** distribuem uma grande quantidade de sensores de forma densa mas sem uma organização em específico. Espera-se que as RSSF não-estruturadas sejam capazes de monitorar ambientes de forma autônoma. Este tipo de organização apesar de dificultar a manutenção da rede, acaba por facilitar sua distribuição inicial.

O outro padrão é chamado de **estruturado**, onde a distribuição inicial ocorre com um pré-planejamento sobre o posicionamento dos nodos. Além disso, este tipo de organização em geral utiliza uma quantidade consideravelmente menor de nodos, facilitando sua manutenção.

Apesar de não se utilizarem de sofisticadas estruturas, as RSSF em geral, fazem uso de três entidades principais, os nodos, os *gateways* e os servidores. A figura 4 ilustra este tipo de arquitetura.

Figura 4 – Arquitetura de uma RSSF



Fonte: do autor.

Dentre os tipos de dados que podem ser coletados através da utilização de RSSF destacam-se:

- Temperatura;
- Umidade;
- Movimento veicular;
- Luminosidade;
- Pressão;
- Presença de componentes químicos;
- Nível de ruído.

A medição destes parâmetros pode ser útil em diversas áreas e aplicações, podendo se citar a saúde, o setor bélico e detecção de desastres naturais.

As redes de sensores sem fio se diferem das redes de computadores tradicionais em diversos aspectos, como, por exemplo, o elevado

número de nodos, a cooperação que deve existir entre eles e as restrições quanto ao consumo de energia (LOUREIRO et al., 2003).

A figura 4 ilustra uma das possíveis configurações de RSSF, onde os nodos sensores coletam informações de um ambiente e as transmitem a um *gateway* que é responsável por um determinado grupo de sensores. Neste caso é possível que não exista a necessidade de se trocar informações entre sensores, porém há ainda os casos onde é necessário que existam trocas de dados entre os nodos a fim de que uma informação seja transportada de um nodo até outro, formando então rotas de transmissão (MARTINS; GUYENNET, 2010; ZIA; ZOMAYA, 2006).

3.2 ASPECTOS DE SEGURANÇA EM RSSF

Faz-se necessário notar que, devido as diferenças de arquitetura entre as redes de computadores e as RSSF, certos aspectos devem ser levados consideração ao se projetar protocolos de segurança para este tipo de rede. Um dos pontos principais, como apontado por Akyildiz et al. (2002), é que enquanto em redes tradicionais se deseja alcançar ótima qualidade de serviço de transmissão, em RSSF o foco é direcionado para a conservação de energia ao custo, muitas vezes, de velocidade de transmissão ou frequência de troca de mensagens.

De acordo com Xiao (2007), a tendência da indústria é que se diminua ainda mais o custo dos sensores sem fio, mantendo, entretanto, a capacidade de armazenamento e de processamento similares as já existentes. Dessa forma, grande parte das pesquisas na área busca otimizar a capacidade de processamento e de armazenamento destes dispositivos mantendo a segurança das comunicações.

Esta seção tem como objetivo apresentar alguns dos aspectos relacionados a segurança em RSSF.

3.2.1 Propriedades de Segurança em RSSF

Como colocado por Padmavathi e Shanmugapriya (2009), as RSSF por conseguirem operar como uma rede *adhoc*, acabam por abordar tanto as propriedades das redes tradicionais, como também propriedades específicas próprias. A união de ambas metas é apresentada na lista abaixo que revisa alguns dos conceitos já apresentados agregando mais informações:

- Confidencialidade: é a habilidade de ocultar informações con-

fidenciais contra agentes mal intencionados na rede, sejam eles nodos participantes ou não;

- Autenticidade: garantir a verificação da identidade do remetente em um ambiente de comunicação aberto e hostil com diversos possíveis nodos comunicantes;

- Integridade: trata da competência de verificar se informações recebidas foram ou não adulteradas, modificadas ou alteradas durante o processo de comunicação;

- Disponibilidade: abrange tanto a questão da obtenção de dados pelos sensores como também o envio e recebimento destes dados na rede;

- *Freshness*: trata da garantia de que a mensagem recebida não é resultado de uma replicação, sendo, portanto, nova na rede;

- Auto Organização: devido a dinamicidade na estrutura de muitas RSSF e a independência que cada nodo possui, a capacidade de auto organização de certas redes é uma necessidade, principalmente em redes onde constantes danos a sensores e ataques de agentes externos ocorrem;

- Sincronização Temporal: é comum que RSSF desejem manter um registro dos atrasos no envio dos pacotes através de rotas de transmissão, necessitando assim de sincronização de tempo. Além disso, existem redes que trabalham de maneira totalmente colaborativa, onde a sincronização dos participantes é obrigatória;

- Localização: devido a, muitas vezes, os dados de um sensor só serem úteis se partirem de sensores vindos de determinadas localizações, verificar este de informação pode vir a ser determinante em certas aplicações.

3.2.2 Limitação de Recursos

Como introduzido previamente, os dispositivos utilizados em RSSF possuem limitado poder de processamento e de armazenamento. Para apresentar uma aproximação quanto a valores de processamento e armazenamento considerados padrão para este tipo de dispositivo, utiliza-se como exemplo o sensor TelosB, dotado de uma CPU de 16-bit de

palavra, 8-MHz de frequência de processamento, 10K bytes de memória *RAM* e 1024K bytes de armazenamento *flash* (ALCARAZ; ROMAN; LOPEZ, 2007; BECHER; BENENSON; DORNSEIF, 2006).

Devido ao fato de que o próprio código fonte dos componentes medidores do dispositivo já ocupam um espaço, é necessário que os algoritmos criptográficos possuam um tamanho reduzido. Um sistema operacional amplamente utilizado em RSSF denominado TinyOS, possui, por exemplo, algo em torno de 4K bytes dependendo da versão utilizada.

O consumo de energia também é um ponto crítico em muitas aplicações, onde o acesso físico ao nodo sensor é dificultado para reposição ou troca de bateria (MARNE; PATIL, 2016; KINGSLY; CHANDRAN, 2013). Protocolos de segurança que necessitam de extensivos cálculos e/ou grandes chaves criptográficas acabam por ter um drástico impacto na vida útil dos sensores.

Wang, Attebury e Ramamurthy (2006), Marne e Patil (2016) trazem à tona a questão do poder de transmissão, explicando que a distância para uma efetiva comunicação entre os nodos é limitada em parte pelo consumo de energia. A situação pode se tornar ainda mais crítica com relação a distância máxima de comunicação quando se trata de ambientes com condições que não favoreçam a transmissão de dados de maneira sem fio.

3.2.3 Ataques em Redes de Sensores Sem Fio

No capítulo anterior foram apresentadas definições relacionadas a área de segurança computacional, onde tipos ataques de segurança de forma geral foram apresentados juntamente com exemplos. Nesta seção o objetivo é a aprofundar tal discussão, direcionando o foco para a área de RSSF.

3.2.3.1 Ataques Físicos

Como as RSSF tratam da distribuição de diversos nodos em locais nem sempre supervisionados, é possível que alguns destes nodos sejam expostos a ataques físicos, podendo estes ataques ser de origem natural ou de agentes externos ao projeto em questão. Vale ressaltar que agentes externos que venham a influenciar negativamente o funcionamento da rede nem sempre tinham tal intenção, podendo ser fruto

de acidentes.

Lopez e Zhou (2008) classificam os ataques físicos quanto a dificuldade de os realizar. A dificuldade, segundo eles, está relacionada ao tempo necessário e aos equipamentos que necessitam ser utilizados durante o processo de ataque. Nota-se, entretanto, que quanto maior a dificuldade, mais danoso é o ataque para o sistema, sendo os ataques mais fáceis os que menos representam desafios para a rede.

Os ataques considerados mais fáceis são aqueles em que o agente mal intencionado consegue interferir nas leituras dos sensores e nas funcionalidades do rádio de transmissão (XU et al., 2006; WANG; ATTEBURY; RAMAMURTHY, 2006; YICK; MUKHERJEE; GHOSAL, 2008). Funcionalidades estas que podem incluir a habilidade de ler, modificar, excluir e até criar mensagens sem que se possua acesso ao algoritmo ou a memória do nodo em questão.

Já o segundo grupo, que possui dificuldade mediana, trata de ataques que fornecem ao agente mal intencionado pelo menos alguma informação relacionada a memória do nodo, seja a memória RAM, memória *flash* ou ainda memória externa (BECHER; BENENSON; DORNSEIF, 2006). Aqui é possível que se obtenha informações como chaves criptográficas por exemplo.

Por fim, o terceiro e mais difícil grupo engloba os ataques que fornecem ao agente mal intencionado controle total sobre o node capturado, o que inclui leitura e escrita na memória, assim como capacidade de analisar o comportamento do software a ele atribuído e realizar alterações baseado em suas necessidades (BECHER; BENENSON; DORNSEIF, 2006).

Existe, entretanto, tecnologias que visam impedir o sucesso de ataques de maior dificuldade como TPM (*Tamper Proof Modules*), porém uma análise mais profunda delas foge do escopo deste trabalho.

3.2.3.2 Ataques Através de Software

Padmavathi e Shanmugapriya (2009), Zia e Zomaya (2006), Rani e Kumar (2017) apresentam tipos de ataques existentes e listam algumas das maneiras de os aplicar. O primeiro grupo é daqueles que envolvem os já descritos anteriormente ataques passivos (monitoramento, espionagem, análise de tráfego, etc).

O segundo grupo por sua vez, que trata de ataques ativos possui algumas peculiaridades quando se trata de RSSF. Um dos ataques aqui está relacionado a manipulação de roteamento. Neste caso, o agente

mal intencionado toma ações como por exemplo, fazer com que muitas das rotas passem por um nodo comprometido, gravando todas as informações transferidas. Outro exemplo, é quando um nodo comprometido apresenta diversas identidades, interferindo assim na maneira como as rotas de transmissão são traçadas.

Outro tipo de ataque é a adição de nodos maliciosos na rede, possibilitando que dados falsos e adulterados sejam transmitidos entre os nodos até a estação central, fazendo com que o resultado obtido das medições seja desvirtuado. Além disso, um nodo malicioso poderia interferir na maneira como as informações são transferidas na rede podendo, por exemplo, se negar a passar dados a diante.

Existe também o ataque quando um nodo malicioso visa replicar um nodo já existente na rede, fazendo com que tanto a questão do roteamento da informação seja desvirtuado, como também a questão de saber de onde a informação realmente veio e se os dados são mesmo legítimos.

3.2.3.3 Pré e Pós Distribuição

Um último tema a ser tratado neste capítulo é a questão das fases de pré e pós distribuição dos sensores nos ambientes. A primeira abrange todas ações que são tomadas em um ambiente diferente de onde os nodos serão dispostos, o qual se sabe que será um ambiente hostil em diversos aspectos. Um ambiente seguro, portanto, é aquele onde se assume que agentes externos não consigam ter acesso a nenhuma das ações que lá acontecem e nem consigam interferir nos processos de tal etapa. Exemplos desta etapa inicial se pode citar a pré distribuição de chaves para os nodos, como explica Yang et al. (2010), Gupta et al. (2013), e o cadastro da identidade dos participantes em bases de dados, como apresenta Herrera e Hu (2012).

A segunda etapa é aquela que compreende o funcionamento da rede após distribuídos os nodos no ambiente a ser monitorado. Nesta etapa, o protocolo deve conseguir lidar com as possíveis adversidades resultantes do ambiente em que os nodos estão inseridos, sejam elas fruto ações de agentes mal intencionados, ou não. É comum em RSSF executar de acordo de chaves baseados nas informações armazenadas em fase de pré distribuição (YANG et al., 2010; GUPTA et al., 2013; SAQIB, 2016).

4 VERIFICAÇÃO FORMAL DE PROTOCOLOS CRIPTOGRÁFICOS

Nesta seção uma visão geral é apresentada sobre o que são protocolos de segurança e os motivos que fazem com que sua verificação formal seja tão importante para a idealização e implementação de sistemas de comunicação de dados efetivamente seguros.

4.1 PROTOCOLOS DE SEGURANÇA

Protocolos de segurança, como descrito por Avalle, Pironti e Sisto (2012), envolvem comunicações de dados e visam proteger informações trocadas por participantes de uma determinada rede apesar da existência de agentes externos constantemente interferindo nas trocas de informações entre participantes.

Como explica Santos (2012), protocolos podem ser classificados conforme suas complexidades. Protocolos de pouca complexidade, chamados de simples, possuem um número reduzido de participantes e mensagens trocadas. Já os protocolos considerados complexos trabalham com elevado fluxo de informações entre diversos participantes, onde os dados devem ser criptografados e devidamente validados.

Como exemplo prático de protocolos simples se pode citar o uso de *Radio-Frequency Identification*, os populares RFIDs utilizados em portarias de condomínios. Já como exemplo de protocolos complexos se pode citar o uso de cartões de crédito, que necessitam cumprir etapas que envolvem diferentes agentes, passando por, pelo menos, o banco, a bandeira do cartão, o consumidor e o responsável pela venda, sendo todos os dados trocados de forma sigilosa.

Além da classificação de protocolos feita acima, como descrito por Santos (2012), os protocolos também podem ser classificados quanto a disponibilidade da sua especificação. Em outras palavras, classifica-se um protocolo como público quando sua especificação for aberta para qualquer um que venha a ter interesse de estudar o funcionamento do protocolo. Já os protocolos privados não disponibilizam sua especificação, oferecendo normalmente somente interfaces de utilização para o usuário final.

4.2 VERIFICAÇÃO FORMAL

Avalle, Pironti e Sisto (2012) comenta que, apesar de o fluxo de execução dos protocolos de segurança ser simples, não é uma tarefa fácil projetá-los sem a presença de falhas. Isso se dá devido ao grande número de variáveis que podem vir a interferir no funcionamento do protocolo, como a violação de regras do protocolo e a falsificação de mensagens.

Como os ataques podem ocorrer em qualquer uma das etapas do protocolo, existe um grande número de maneiras para se tentar quebrar a segurança dos protocolos. Apesar de existirem boas práticas e recomendações sobre como proceder ao se projetar um protocolo de segurança, como apresenta Abadi e Needham (1994), a modelagem e a implementação deste tipo de protocolo ainda podem apresentar falhas de projeto.

Buscando eliminar o número de falhas de projeto dos protocolos de segurança se recomenda a utilização de métodos lógicos e matemáticos durante seu desenvolvimento. O uso deste métodos atualmente é feito através de um conceito denominado verificação formal de protocolos criptográficos, definição que surgiu, como mostra Abadi (2006), no início de 1980 e teve sua maior maturação durante a década seguinte.

Existem propostas de como realizar este tipo de verificação formal. Nogueira (2008) explica a diferença entre cinco destas propostas de modelagem. A seguir serão apresentados dados sobre quatro destes tipos individualmente com caráter informativo, porém o foco maior se dá em Sistemas Especialistas por ser a modelagem base deste trabalho.

4.2.1 Ferramentas de Propósito Geral

Neste tipo de verificação, técnicas como a utilização de máquinas de estados finitos para modelar os protocolos são as mais comuns. Aqui o envio ou recebimento de mensagens resulta em mudanças de estado no grafo. A verificação do protocolo portanto se dá através da análise da máquina de estados formada pelos participantes do protocolo. Utiliza-se uma técnica chamada de **análise de alcançabilidade** para provar a correção com relação a sua especificação, porém este tipo de técnica não é capaz de provar nada sobre a atuação de uma adversário ativo sobre o protocolo (NOGUEIRA, 2008).

4.2.2 Lógicas de Análise de Conhecimento e Confiança

Este tipo de abordagem faz uso do que se conhece como lógica modal, lógica essa que faz uso de uma linguagem que descreve declarações sobre confiança, conhecimento de mensagens e regras de referência. A mais conhecida aplicação deste tipo de abordagem é a lógica BAN . O objetivo é que se encontre, ao final da análise, declarações que representem condições corretas de um protocolo (NOGUEIRA, 2008). Em um protocolo de autenticação mútua, por exemplo, deseja-se que participantes consigam obter provas irrefutáveis de que ambos participaram do processo (PIVA, 2009).

A lógica BAN foi pioneira em apresentar uma sintaxe lógica própria, obtendo grande adesão da comunidade devido a seus postulados serem intuitivos e permitirem análises diretas. Entretanto, este tipo de abordagem peca ao não dar suporte a modelagem de certas características especiais dos protocolos criptográficos, como resumos criptográficos e acordo de chaves.

4.2.3 Sistemas Especialistas

Diferentemente do que ocorre com a análise através da utilização de máquinas de estados finitos, em sistemas especialistas o ambiente ao qual o protocolo é executado já inicia indesejado. Aqui os adversários possuem diversas funcionalidades ativas. Se deseja provar que um protocolo, partindo de um estado inicial, consegue alcançar um estado final almejado sem que agentes externos mal intencionados consigam interferir na comunicação. Este tipo de abordagem se divide em dois grandes grupos:

4.2.3.1 Modelos Simbólicos

O primeiro grupo teve suas pesquisas iniciadas com Needham e Schroeder (1978), onde um modelo para representar as capacidade que um atacante possui sobre o funcionamento de um protocolo foi proposto. O atacante era capaz de, como descrito por Santos (2012) e Avalle, Pironti e Sisto (2012):

- Permear-se entre a comunicação de dois participantes em qualquer processo do protocolo;

- Modificar e copiar fragmentos das informações enviadas na rede;
- Replicar mensagens;
- Forjar mensagens.

O atacante neste modelo era incapaz de descobrir as chaves utilizando força bruta (forçar inúmeras vezes diversas combinações de chave para até descobrir a verdadeira). Além disso, também era incapaz de analisar criptograficamente as mensagens em busca de informações que levassem ao descobrimento da chave utilizada.

Apesar de tal modelo contemplar boa parte das possíveis ações de um participante desonesto, alguns pontos ainda não haviam sido abordados. Foi através do trabalho de Dolev e Yao (1983) que se expandiu as capacidade do atacante, sendo este o modelo simbólico que mais se utiliza na segunda década dos anos 2000. Neste novo modelo, que leva o nome de seus criadores Dolev-Yao, o atacante é capaz então, além das ações descritas anteriormente, de:

- Manter registro de todas as mensagens transmitidas na rede;
- Tornar-se um participante efetivo da rede, iniciando troca de mensagens de forma direta com outros participantes;
- Receber respostas a mensagens enviadas a outros participantes.

Apesar de o modelo Dolev-Yao fazer parte do estado da arte quanto a modelagem simbólica formal de protocolos de segurança, existem ainda estudos sendo feitos na área, visto que, o avanço acelerado do surgimento de novas tecnologias dinamiza como as ameaças podem vir a surgir.

Dessa forma, novas ameaças podem fazer parte das verificações formais atuais, evitando que casos como o descrito por Needham e Schroeder (1978) voltem a ocorrer. O caso deste protocolo ganhou fama devido ao fato de que, por dezessete anos seu fluxo de execução foi considerado seguro, até que Lowe (1995) mostra a quais falhas que o protocolo estava exposto ao submetê-lo a uma verificação formal.

Neste tipo de modelo, como o nome sugere, as informações são representadas através de símbolos e as funções criptográficas representadas através da álgebras destes símbolos. Por exemplo, caso se deseje representar a cifragem simétrica de uma determinada mensagem m com uma chave k qualquer, define-se a função (ou construtor muitas vezes chamado)

$$encrypt(m, k)$$

onde *encrypt* é um nome arbitrário dado a função que realiza a cifragem do primeiro parâmetro fornecido por uma chave presente no segundo parâmetro. Já para modelar a decodificação simétrica, modela-se a função de tal maneira que somente se é possível obter a mensagem original estando em posse da mensagem criptografada e da chave que a codificou, tendo, por exemplo, o seguinte formato

$$decrypt(encrypt(m, k), k)$$

onde *decrypt* é um nome arbitrário dado a função (ou destrutor) que realiza a decodificação do primeiro parâmetro fornecido por uma chave presente no segundo parâmetro.

4.2.3.2 Modelos Computacionais

Por serem menos abstratos do que os modelos simbólicos, como descrito por Avalle, Pironti e Sisto (2012), os modelos computacionais são mais próximos da realidade. Neste tipo de modelo, os dados são representados como cadeias de *bits* enquanto as primitivas criptográficas (criptografia de chave simétrica, função de resumo criptográfico e criptografia de chave pública (ALCARAZ; ROMAN; LOPEZ, 2007)) são representadas através de algorítimos probabilísticos polinomiais de tempo aplicados em cadeias de *bits*.

Assume-se que os algorítimos sejam capazes de modelar tanto funcionalidades positivas nos protocolos ($decrypt(encrypt(m, k), k) = m$, por exemplo), assim como também, modelar as propriedades negativas dos protocolos (distinguir entre duas implementações do esquema de criptografia). Agentes mal intencionados são representados como qualquer algoritmo polinomial de tempo que possua acesso a canais públicos de comunicação no protocolo.

Os agentes mal intencionados também são dotados de certos oráculos, conceito este introduzido por Bellare e Rogaway (1993) que modela funções das quais não se conhece o funcionamento interno, mas que entretanto, dependendo do valor de entrada, produz um resultado independente diferente, mantendo um histórico das entradas já utilizadas para caso sejam novamente necessárias). Oráculos são utilizados aqui para modelar certas informações extras que atacantes talvez tenham acesso.

Os protocolos no modelo computacional são representados como máquinas probabilísticas, sendo as propriedades de segurança apenas garantidas quando a probabilidade de suas propriedades serem violadas ao se executar a máquina probabilística for desprezível (AVALLE; PIRONTI; SISTO, 2012).

4.2.4 Outras Abordagens

Além das técnicas já apresentadas acima, existem ainda outras alternativas mais atuais que buscam tratar as diversas interações entre protocolos que existem nas transações de informação das redes tradicionais. Este tipo de verificação se faz necessária devido a grande parte das vulnerabilidades se concentrarem na transição de informações de um protocolo para outro, ou seja, nos limites de onde acaba a atividade de um protocolo e inicia a atuação de outro. Este tipo de cenário é comum onde se exista a troca de informações entre diferentes organizações envolvidas em transações.

5 PROTOCOLOS VERIFICADOS

Nesta seção são apresentados e explicados os três protocolos que foram verificados formalmente por este trabalho.

5.1 TINY PBC

O primeiro protocolo de segurança analisado por este trabalho é o *Tiny PBC*, descrito por Oliveira et al. (2008). Seus autores o descrevem como capaz de realizar emparelhamentos entre os nodos de forma autenticada através da utilização de identificadores sem a necessidade haver a necessidade de troca de chaves e, mesmo assim, ser capaz de fazer com que pares de nodos compartilhem uma chave em comum não previamente armazenada. A criptografia do *Tiny PBC* se baseia em duas das técnicas estudadas anteriormente: criptografia de curva elíptica e pareamento bilinear.

O acordo de chaves proposto pelo protocolo não exige que os participantes troquem um determinado número de mensagens para que consigam chegar a uma chave simétrica comum. Este tipo de abordagem é chamada na literatura de distribuição de chaves não interativa. A autenticação se dá através das informações que identificam cada nodo, sendo neste caso um identificador individual. Estes identificadores podem ser quaisquer informações que especifiquem o indivíduo para os participantes da rede.

O protocolo é dividido essencialmente em duas etapas, uma ocorre de maneira *off-line* antes da distribuição dos sensores na área desejada e a outra ocorre após os sensores já estarem no ambiente hostil. Na primeira etapa, uma Autoridade Certificadora escolhe uma chave, chamada de chave mestra. A chave mestra será representada pelo símbolo **S**.

Após feita a escolha de **S**, a Autoridade Certificadora é responsável também por mapear cada um dos identificadores $(\boldsymbol{id_x})$ para um ponto da curva elíptica através de uma função *hash* chamada de ϕ, sendo este ponto chamado de $\boldsymbol{P_x}$ e ser também a chave pública a ser atribuída a cada um dos respectivos nodos.

$$P_x = \phi(id_x)$$

Calculada a chave pública de cada um dos nodos, a Autoridade Certificadora agora calcula cada uma das chaves privadas dos nodos

através da equação

$$S_x = [\mathbf{S}]P_x$$

Terminado o cálculo das chaves privadas, a Autoridade Certificadora carrega de forma *off-line*, em um local assumido como seguro, ou seja, sem a possibilidade de interferência de agentes externos, a chave privada e o **id** de cada nodo, juntamente com a função ϕ, capaz de calcular uma chave pública através de um **id** qualquer.

Após os nodos possuírem suas respectivas chaves privadas, os dispositivos podem então ser distribuídos nos ambientes hostis no qual dados serão coletados. Neste ambiente, os nodos começam a segunda fase do protocolo, que descreve que os nodos iniciam então uma descoberta dos nodos vizinhos e, consequentemente, dos seus respectivos **id**.

Sabendo agora o **id** dos dispositivos vizinhos, um determinado nodo pode tentar iniciar uma comunicação segura enviando uma mensagem criptografada utilizando uma chave calculada. Como:

$$K_{i,j} = \hat{e}(S_i, P_j)$$

O diagrama da Figura 5 apresenta um exemplo de como ocorreria o fluxo do protocolo em questão quando dois nodos, chamados de **A** e **B** desejam iniciar uma comunicação segura.

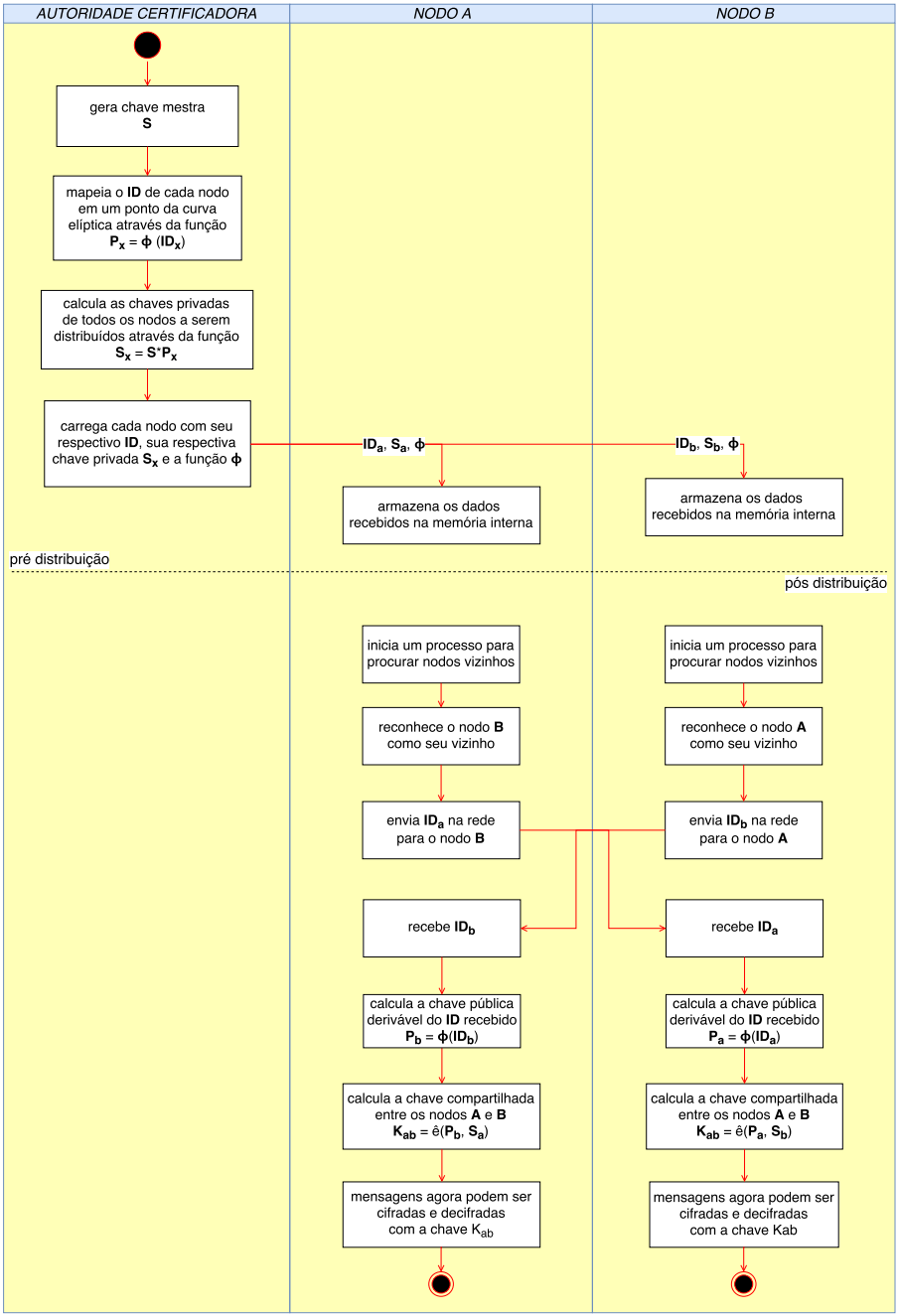


Figura 5 – Diagrama de Fluxo - TinyPBC

5.2 ADRIAN HERRERA

O protocolo apresentado por Herrera e Hu (2012) propõe um modelo de distribuição de chaves entre participantes de uma rede que possua uma estrutura hierárquica de até um nível de diferença. Neste tipo de estrutura, o participante pode fazer o papel ou de nodo coletor de dados ou de estação base, responsável por receber informações dos coletores.

O objetivo deste protocolo é fazer com que, ao fim do processo de acordo de chaves, tanto a estação base como o nodo que deu início ao protocolo possuam uma chave simétrica a ser utilizada na cifragem das futuras trocas de informação.

O protocolo inicia escolhendo as chaves privadas e públicas de cada um dos nodos participantes da rede. Após feitas as escolhas, a estação base envia estas chaves para os respectivos nodos juntamente com os seus **ids**. A própria chave pública da estação base também é enviada. Estes processos inicias ocorrem no que foi definido anteriormente como ambiente de pré distribuição, um ambiente seguro no qual agentes externos não possuem acesso e são incapazes de comprometer qualquer dos processos que ali ocorrem.

A segunda etapa deste protocolo, por sua vez, ocorre após a distribuição dos nodos no ambiente hostil, onde os participantes estão vulneráveis a agentes externos. Nesta fase, os nodos devem enviar uma mensagem à estação base solicitando a chave compartilhada entre ambas as entidades. Tal pedido deve conter a identificação do nodo dada pelo **id** e um ***nonce*** (valor arbitrário a ser utilizado uma única vez). Enquanto o **id** tem como objetivo informar à base a quem a chave simétrica será endereçada, o ***nonce*** tem como função garantir que a requisição recebida não é fruto de uma repetição de dados.

Aqui os autores do protocolo não especificam se os ***nonce*** são únicos para cada nodo ou para a rede como um todo. Para tratar esta questão, ao se verificar formalmente o protocolo, algumas premissas são propostas no próximo capítulo, dentre elas, que cada ***nonce*** é associado ao nodo que primeiro o enviou, sendo possível sua repetição em diferentes nodos.

Assume-se também que, ao identificar a repetição de um ***nonce***, a estação base aborta o fluxo de execução para o nodo em questão. Assumindo que cada ***nonce*** é associado apenas ao nodo (e não à rede como um todo), modela o pior dos casos, onde um agente externo pode falsificar uma requerimento utilizando um ***nonce*** previamente enviado na rede.

Em posse de uma requisição válida, a estação base gera a chave simétrica a ser utilizada por ambos os participantes. Esta chave passa por dois processos antes de ser efetivamente enviada na rede. O primeiro deles é dado pela cifragem da chave através da utilização da chave pública do nodo requerente. O segundo processo, por sua vez, é responsável por obter um valor *hash* da chave simétrica e o assinar digitalmente utilizando a chave privada da estação base. Finalizados ambos os processos, os valores são então concatenados e enviados na rede aberta para o nodo solicitante.

Ao receber as informações cifradas pela estação base, o nodo primeiramente tenta decifrar o segmento da mensagem que contém a chave secreta através da utilização sua própria chave privada (seguindo assim o esquema de chave pública normalmente aplicado). Feito isso, o nodo utiliza a mesma função *hash* utilizada pela estação base na chave recém descriptografada. O resultado de tal *hash* deve ser igual ao valor encontrado ao se verificar a assinatura digital, que é resultado da descriptografia do outro segmento da mensagem recebida utilizando a chave pública da estação base (armazenada durante a primeira fase deste protocolo).

Caso os valores coincidam, a integridade da mensagem está garantida e, segundo os autores, a autenticidade também, portanto o nodo envia uma mensagem de *acknowledge* para a estação. Caso contrário, o procedimento é abortado devido ao indício de adulteração do protocolo

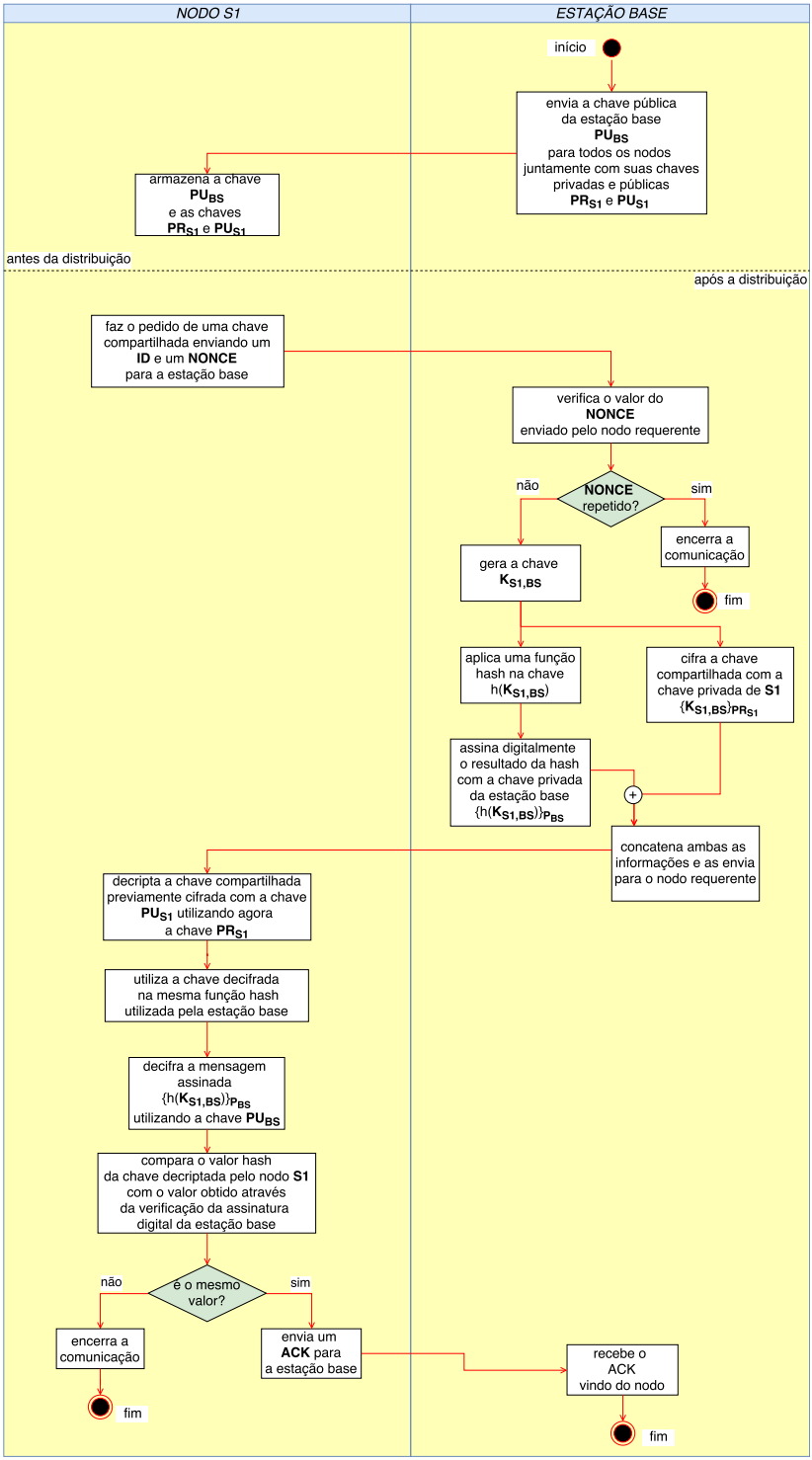


Figura 6 – Diagrama de Fluxo - Herrera

5.3 NAJMUS SAQIB

Saqib (2016) descreve um protocolo para a troca de chaves que visa suprir algumas das vulnerabilidades apresentadas por protocolos como o ECDH (BONEH; SHPARLINSKI, 2001). Este protocolo aborda a interação entre dois nodos participantes de mesma hierarquia, sendo ambos coletores de dados. Assim como os outros protocolos vistos anteriormente, o objetivo é fazer com que, ao fim do acordo de chaves, ambos os participantes estejam de posse de uma chave simétrica a ser utilizada em comunicações futuras entre os dois participantes.

O protocolo tem início em uma fase de pré distribuição, onde uma estação base envia para cada um dos nodos a serem distribuídos a equação que descreve a curva elíptica a ser utilizada, juntamente com o ponto de partida **G** a ser utilizado nas operações. Além disso, as chaves privadas e públicas de cada nodo também são enviadas, sendo a chave privada dada por $\boldsymbol{K_i}$ (onde **i** é um índice atribuído a cada nodo) e a chave pública dada por $\boldsymbol{K_i}.\mathbf{G}$.

Após armazenadas as informações acima em cada um dos participantes da rede e tendo os nodos distribuídos no ambiente a ser monitoriado, a segunda fase do protocolo pode ter início. Aqui, um nodo que necessite trocar algum tipo de informação com um vizinho deve cumprir os procedimentos descritos a seguir.

Primeiramente o nodo deve escolher um número aleatório qualquer, chamado aqui de **X**, e, a partir dele, calcular uma chave secreta dada pela multiplicação escalar **X.G**. Feito isso, o nodo então cifra a soma do valor da sua chave privada com o valor **X** previamente escolhido, utilizando a chave pública do nodo com que necessita realizar a troca de informações. O processo de cifragem é representável da seguinte maneira:

$$(K_i + X).PU_j$$

O valor resultante é então enviado ao nodo **j**. Ao receber a mensagem, o nodo **j** por sua vez aplica a seguinte equação para descriptografar a chave secreta enviada por **i**

$$((K_i + X).PU_j).{K_j}^{-1}) - PU_i = X.G$$

Expandindo a equação acima, tem-se que

$$((K_i + X)K_j.G).{K_j}^{-1}) - K_i.G = X.G$$

Simplificando os termos, obtém-se

$$((K_i + X).G) - K_i.G = X.G$$

Expandindo a equação acima, tem-se

$$(K_i.G + X.G) - K_i.G = X.G$$

Por fim, encontra-se que

$$X.G = X.G$$

A chave simétrica deste protocolo é composta pela soma de duas chaves secretas. O nodo **i** já possui uma, dada por **X.G**, necessitando da segunda, que será fornecida pelo nodo **j**. O nodo **j**, por sua vez, acaba de descriptografar a chave enviada pelo nodo **i**, necessitando agora criar a sua própria. Tal criação se dá de forma similar a do seu nodo vizinho, sendo portanto escolhido um valor **Y** qualquer e sendo a chave secreta de **j** resultado da multiplicação escalar de **Y.G**.

A chave simétrica portanto é dada pela equação

$$K_{ij} = (XG + YG)$$

O processo de envio da chave secreta do nodo **j** para o nodo **i** se dá de forma similar ao que já foi visto anteriormente. Aqui o nodo **j** cifra a soma da sua chave privada com o valor **Y** escolhido, utilizando a chave pública do nodo **i**. O processo de cifragem é representável da seguinte maneira:

$$(K_j + Y).PU_i$$

Já a descriptografia de tal mensagem pelo nodo **i** é dada por

$$((K_j + Y).PU_i).{K_i}^{-1}) - PU_j = Y.G$$

Reduzindo a equação de forma similar a realizada anteriormente, obtém-se

$$(K_j.G + Y.G) - K_j.G = Y.G$$

A Figura 7 ilustra todas as etapas do protocolo descritas acima.

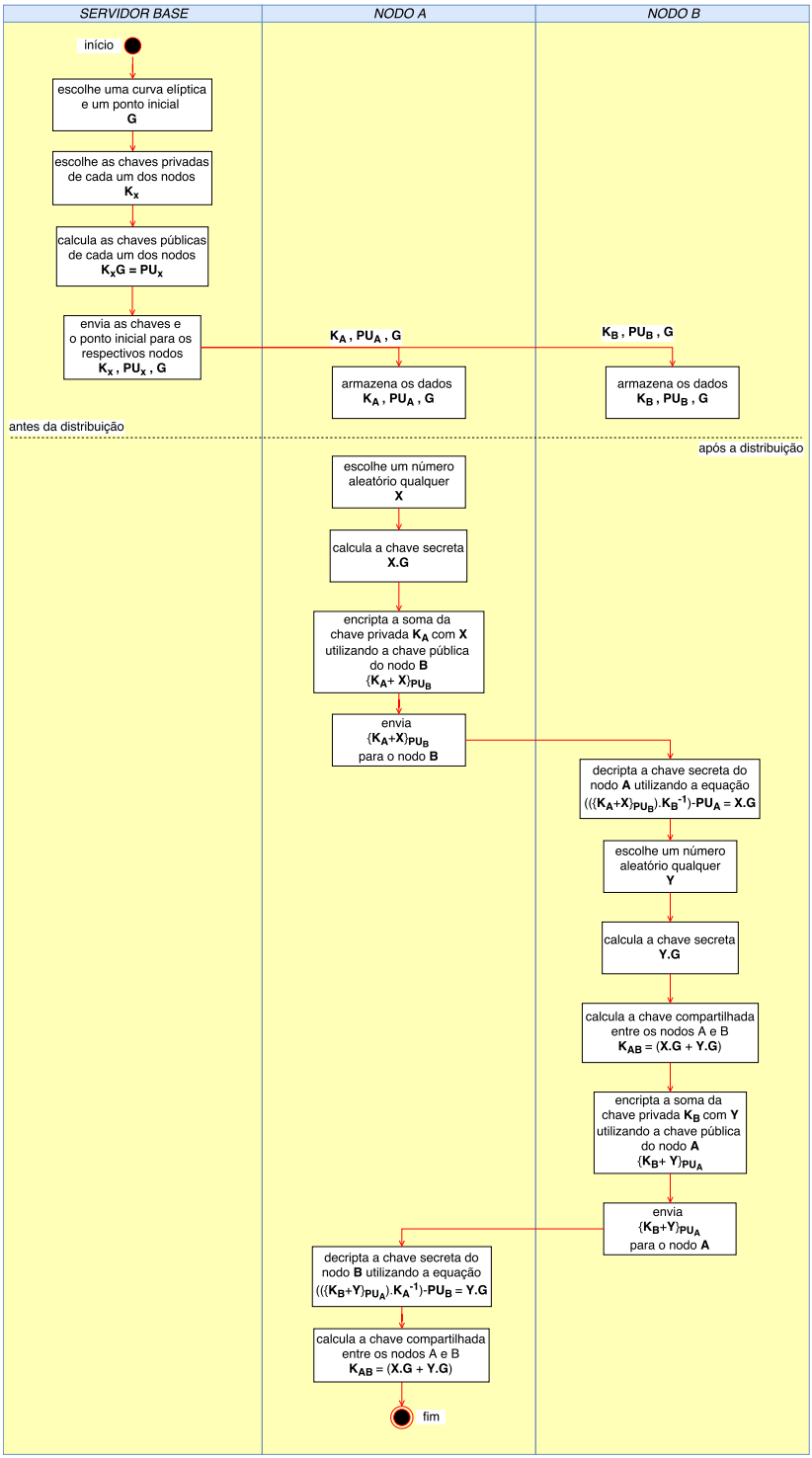


Figura 7 – Diagrama de Fluxo - Najmus Saqib

6 VERIFICAÇÃO DOS PROTOCOLOS

Neste capítulo, são apresentadas as verificações formais realizadas nos protocolos descritos no capítulo anterior juntamente com uma apresentação sobre a ferramenta utilizada, chamada de ProVerif.

6.1 PROVERIF

O ProVerif é, como descrito em sua página oficial (BLANCHET, 2017), um verificador de protocolos criptográficos simbólico baseado no modelo formal, chamado de Dolev-Yao, como visto nos capítulos anteriores. Este verificador é capaz de lidar com muitas das primitivas criptográficas utilizadas no processo de criação de protocolos criptográficos.

A escolha do ProVerif como ferramenta a ser utilizada para o desenvolvimento deste trabalho se deu devido a detalhada documentação fornecida pelos desenvolvedores e a comunidade que a utiliza. São diversos os exemplos de publicações que comprovam a eficácia da ferramenta na verificação formal de protocolos criptográficos, independentemente da aplicação dos mesmos Nguyen, Oualha e Laurent (2016), Hirschi, Baelde e Delaune (2016), Diaz, Arroyo e Rodriguez (2014), Kleberger e Moulin (2013), Delaune et al. (2010), Vigo e Filè (2009), Delaune, Kremer e Ryan (2009), Shen et al. (2014), Arapinis, Cheval e Delaune (2015).

O ProVerif, como explica Blanchet et al. (2017), aceita arquivos codificados em diversas linguagens de entrada, entretanto, a considerada estado da arte para a ferramenta (e preferencial) é a *Applied Pi Calculus*, como explica Ryan e Smyth (2011), uma extensão da linguagem Pi Calculus (ou π Calculus), apresentada originalmente por Milner, Parrow e Walker (1992).

Applied Pi Calculus é uma linguagem desenvolvida especificamente para descrever as interações entre processos concorrentes, sendo aplicada principalmente na descrição e na análise de protocolos de segurança. A linguagem *Applied Pi Calculus* se difenrencia da sua linguagem mãe devido ao rico algebrismo que se pode utilizar na modelagem dos protocolos. Esta linguagem permite que o desenvolvedor expresse objetivos de segurança a serem alcançados pelos protocolos e se os objetivos são ou não alcançados durante sua execução.

No ProVerif, o papel de cada participante dentro do protocolo é

modelado através de processos. Estes processos são executados paralelamente e se pode definir que ilimitadas instâncias de cada processo sejam executadas. A ferramenta traduz então os processos para cláusulas de Horn, apresentadas por Angluin, Frazier e Pitt (1992), que representam de forma abstrata o protocolo. A análise formal então ocorre quando a ferramenta recebe como entrada as cláusulas derivadas e analisa que fatos um agente mal intencionado é capaz de aprender sobre a execução do protocolo em questão.

As propriedades de segurança a serem verificadas são modeladas como consultas de derivabilidade. Por exemplo, para provar o sigilo de um termo $\boldsymbol{k}$ dentro do protocolo, a ferramenta tenta provar que, ao fim da execução, $\boldsymbol{k}$ não faz parte do conjunto conhecimento de agentes mal intencionados.

A modelagem de protocolos no ProVerif se dá através de termos gerados a partir de tipos de dados, variáveis, nomes, construtores e destrutores. Estes dois últimos são utilizados para a manipulação dos dados. Suas prototipagens já foram apresentadas neste documento na seção 4.2.3.1, onde se apresentou como as funções dos modelos simbólicos são representadas e como elas manipulam os dados a elas fornecidos. Mais informações sobre como canais de comunicação, termos reservados, nomes privados, entre outras especificações da ferramenta podem ser encontradas em (BLANCHET et al., 2017).

A ferramenta pode apresentar três possíveis resultados para consultas a ela fornecidas:

- *Verdadeiro*: neste caso, a ferramenta consegue provar que não existe nenhum ataque com relação a indagação fornecida. Aqui nenhum tipo de rastro do ataque é apresentado pela ferramenta, apenas as cláusulas que foram utilizadas para formulação da prova;

- *Falso*: a consulta fornecida apresenta um ataque à propriedade de segurança desejada, não sendo possível provar sua corretude. O ProVerif então fornece uma descrição do traço do ataque que ocorre no protocolo;

- *Não pode ser provado:* por fim, este tipo de saída é resultante de uma análise que não consegue nem provar que a propriedade é assegurada e nem consegue definir um ataque que descreva porque motivos a propriedade não é provada correta. Contudo, a ferramenta é capaz de trazer traços de possíveis ataques e provas de teoremas que podem vir a auxiliar na verificação formal a ser realizada.

6.1.1 Sigilo e Autenticidade

Este trabalho tem como objetivo avaliar a garantia (ou não) de duas das mais importantes propriedades de segurança: sigilo e autenticação. Devido à limitações da ferramenta utilizada, somente estas duas propriedades foram verificadas.

Blanchet (2016) explica que a propriedade de sigilo é alcançada, na prática, quando um agente não autorizado não consegue obter informações essenciais relacionadas aos dados sendo manipulados pelo protocolo.

Já a definição prática de autenticação é apresentada de maneira formal pioneiramente por Woo e Lam (1993), onde os autores definem autenticação como a certeza que uma entidade possui sobre a identidade de uma outra entidade autenticada. Aplicando este conceito em protocolos, isso significa que, ao término de uma determinada etapa do processo uma entidade deve ter certeza (ou seja, existem informações o suficiente que comprovem isso) de que a comunicação realizada foi de fato efetuada com quem se acredita ter sido realizada.

Posteriormente, Lowe (1997) dá continuidade ao trabalho de Woo e Lam, detalhando hierarquias existentes quando se trata de autenticação. O autor apresenta a dificuldade que se tem na literatura acadêmica em se chegar a um consenso quanto a definição de autenticação. Lowe descreve com o objetivo de formalizar a autenticação, quatro níveis de autenticação:

6.1.1.1 Vivacidade

Vivacidade é considerada a definição de autenticação mais fraca dentre as existentes. Diz-se que um protocolo garante a um agente iniciador de comunicação **A** a vivacidade de um agente **B** se, sempre que **A** completar etapas do protocolo, aparentemente com **B**, então **B** esteve em algum momento também executando este protocolo.

Vale destacar que, **B** não necessariamente precisa ter acreditado que estava se comunicando com **A**.

6.1.1.2 Acordo Fraco

Fortalece-se o grau de segurança proposto no nível anterior insistindo que **B** tenha concordado em executar o protocolo com **A**. Por

definição, o protocolo garante a um agente **A**, atuando como iniciador, um acordo fraco com um agente **B** se, sempre que **A** atuando como iniciador, completar etapas do protocolo aparentemente com **B**, então **B** previamente executou o protocolo aparentemente com **A**.

Vale ressaltar que **B** não necessariamente atuou durante toda a execução do protocolo de forma responsiva.

6.1.1.3 Acordo Não Injetável

Estreitando ainda mais a relação entre os participantes, neste nível se adiciona a condição de que ambos os participantes concordam com relação ao papel de cada um dentro do protocolo além de concordarem com certos dados utilizados na troca de informações.

A definição formal deste tipo de autenticação traz que o protocolo garante a um agente **A**, atuando como um iniciador, um acordo não injetável com um agente **B**, que atua como um respondedor, que trata de um conjunto de informações X (onde X engloba variáveis apresentadas na descrição do protocolo) se, sempre que **A**, atuando como iniciador, concluir a execução do protocolo, aparentemente com o respondedor **B**, então **B** previamente já esteve executando o protocolo aparentemente com o agente **A** e **B** estava atuando como um respondedor em seu turno e ambos os agentes concordaram com os valores correspondentes as variáveis presentes em X.

Tal definição, entretanto, não garante uma relação de um para um entre as etapas executadas por **A** e as etapas executadas por **B**.

6.1.1.4 Acordo Injetável

Por fim, neste tipo de autenticação se deseja evidenciar a existência de uma relação de um para um entre os participantes.

A definição traz que um agente **A**, atuando como iniciador, possui um acordo com um agente **B**, que atua como respondedor, relacionado a um conjunto de informações X se sempre que **A**, atuando como iniciador, conclui a execução do protocolo aparentemente com o respondedor **B**, então **B** estava anteriormente executando o protocolo aparentemente com **A** e **B** estava atuando como respondedor em seu turno, sendo que, ambos os agentes concordaram nos valores correspondentes ao conjunto de informações X e que cada execução da parte de **A** corresponde a uma única execução de **B**.

6.1.1.5 Recenticidade

Um outro termo importante que pode vir a ser crucial no processo de autenticação de certos protocolos é a recenticidade das informações durante a execução dos protocolos. A recenticidade em protocolos de segurança está relacionada a quão recente as mensagens trocadas entre os participantes são. A interpretação do termo recente varia entre protocolos, podendo, por exemplo, ser tratada como um valor definido de unidades de tempo antes do término de uma etapa do protocolo ou ainda dentro do tempo de duração de execução de uma etapa do protocolo. Fica a cargo dos desenvolvedores especificar o que lhes é mais conveniente.

Entretanto, é necessário que as mensagens trocadas entre os participantes possuam algum tipo de informação que identifique em que momento a informação foi enviada, deixando claro que não foi em um tempo qualquer do passado, mas sim em um valor de tempo claramente definido.

Os níveis de autenticação acima descritos não tratam de recenticidade por definição, entretanto todos eles podem ser modificados para que tratem deste tipo de análise. Protocolos que se utilizam de verificações de recenticidade tendem, em geral, a garantirem níveis maiores de segurança quando comparados a aqueles que não o fazem. Deve-se ressaltar que este tipo de informação temporal acaba por aumentar o volume de dados durate a troca de mensagens.

6.2 RESULTADOS

Após a modelagem dos protocolos apresentados no capítulo anterior em *Applied Pi Calculus*, os modelos foram submetidos ao ProVerif, gerando assim um montante de informações que foram utilizadas para identificar se as propriedades de segurança de sigilo e autenticação (especificando em qual nível se encontra) são (ou não) garantidas pelo protocolo nais quais elas foram formalmente verificadas. Quanto ao nível de autenticação, verificou-se apenas os casos de não injetividade e injetividade, sendo vivacidade e acordo fraco desconsiderados dos resultados.

Esta seção, portanto, apresenta premissas adotadas para a verificação formal, os métodos utilizados e as provas por eles alcançadas. Destaca-se que este capítulo objetiva trazer os pontos principais, na visão do autor, dos resultados obtidos através das verificações formais.

As validações na íntegra encontram-se nos arquivos separados anexados a este documento. No entanto, recomenda-se que o leitor acompanhe a apresentação dos resultados das verificações formais juntamente com os arquivos anexado, facilitando a visualização das etapas aqui explicadas.

6.2.1 TinyPBC

A análise do protocolo TinyPBC se baseia integralmente nos artigos de Oliveira et al. (2008) e Oliveira et al. (2011), escrito pelos desenvolvedores e idealizadores do protocolo em questão. Algumas premissas foram estabelecidas para a adequação do protocolo ao modelo proposto a ser analisado pelo ProVerif. São elas:

- A primeira premissa traz que agentes externos não possuem acesso físico aos nodos e estações base, sendo impossível a adulteração ou captura física aos dispositivos;
- A segunda premissa trata da questão de como os nodos tem acesso aos IDs dos seus vizinhos. Os autores descrevem em ambas as publicações que é razoável assumir que os nodos conhecem os **ids** de seus vizinhos devido a questão do roteamento dos dados até a estação base. Visto que os autores também descrevem no artigo que novos nodos podem ser adicionados a rede sem problemas, é então coerente assumir que os nodos em algum momento divulgam seus **ids** abertamente;
- A terceira premissa trata da questão da recenticidade das informações, onde, baseado no texto, não existe nenhuma indicação de tempo na troca de mensagens entre os nodos. Não podendo assim, levar-se em conta a questão da recenticidade na análise da autenticidade;
- A quarta premissa adotada traz que o protocolo já finalizou a fase de pré distribuição dos nodos e tem seu início em estado de pós distribuição, já em ambiente hostil;
- A quinta premissa diz que informações atribuídas aos nodos em fase de pré distribuição não estão inicialmente disponíveis aos agentes mal intencionados. Em outras palavras, apesar de os dados não estarem no banco de informações inicial dos agentes externos, isso não impede que, durante o fluxo de execução do protocolo, estas informações não sejam inferidas pelo / enviadas para agentes externos.

- A sexta premissa traz que: para que se consiga provar a autenticidade dos participantes da comunicação, estende-se o protocolo até uma fase de troca de mensagens. Esta medida é necessária em vista do caráter não interativo da distribuição de chaves entre os nodos.

Partindo das premissas acima, modelou-se o protocolo em questão em *Applied Pi Calculus*. O código do fonte do modelo é destrinchado e explicado abaixo. O cenário modelado para as verificações trata da troca de mensagens entre dois nodos quaisquer denominados arbitrariamente no código da modelagem como **A** e **B**.

Como já explicado nas premissas, este cenário se passa após a distribuição dos nodos, o que faz com que a autoridade certificadora não tenha participação na troca de mensagens entre os nodos. Aqui, inicialmente, se é apresentado um cenário onde o nodo **A** deseja se comunicar com o nodo **B**.

Código 1 – TinyPBC: Tipos de Dados

```
1  type mkey.
2  type g.
3  type id.
```

Em Código 1 são definidos os tipos de dados a serem tratados durante o fluxo de execução do protocolo. O tipo **mkey**, descrito na linha {*1*}, é utilizado para representar as chaves mestras criadas. Já **g** {*2*}, por sua vez, representa dados relacionados a curva elíptica. Por fim, **id** {*3*} simboliza os identificadores dos nodos participantes da rede.

Código 2 – TinyPBC: Declaração de Nomes Livres e Constantes

```
1  free S:mkey [private].
2  free C:channel.
3  const ELIPTICCURVE: g [private].
4  free IDA: id [private].
5  free IDB: id [private].
```

Já em Código 2 são tratadas as declarações dos nomes livre (*free names*) e das constantes a serem trabalhadas. Como já explicado no capítulo anterior, **S** é o nome atribuído à chave mestra {*1*}. O canal onde ocorrem as comunicações é chamado de **C** {*2*}. Define-se a curva elíptica como uma constante chamada de **ELIPTICCURVE** {*3*}. Por fim, são declarados os **IDs** dos participantes deste cenário, representando respectivamente os nodos **A** {*4*} e **B** {*5*}.

Código 3 – TinyPBC: Construtores e Destrutores

```
1  fun phi(g, id): g.
2
3  fun calcPrivateKey(g, mkey): g.
4
5  fun calcPairKey(g,g): g.
6
7  fun encrypt(g, bitstring): bitstring.
8
9  reduc forall M: bitstring , K: g ; decrypt ( encrypt(K, M) , K ) = M.
10
11 equation forall        PA: g, PB: g, SKEY:
12     mkey; calcPairKey( calcPrivateKey(PA, SKEY), PB )
13         =  calcPairKey( calcPrivateKey(PB, SKEY), PA ) [linear].
```

Em Código 3 são apresentados os construtores e destrutores responsáveis pela manipulação dos dados. Na linha {*1*} é definido o construtor que, como apresentado no capítulo anterior, é responsável por associar **ids** à pontos da curva elíptica escolhida para o protocolo. Já o construtor modelado na linha {*3*} objetiva calcular as chaves privadas de cada nodo ao receber um ponto na curva elíptica.

O construtor descrito na linha {*5*} é a representação de como ocorre o cálculo da chave simétrica a ser utilizada pelos nodos ao fim do acordo de chaves. O construtor definido na linha {*7*} é a representação da cifragem de uma *string* de *bits* utilizando uma chave qualquer representada por um ponto na curva elíptica, sendo seu destrutor apresentado na linha {*9*}. Por fim, a equação descrita entre as linhas {*11*} e {*13*} tem como objetivo modelar a propriedade específica dos pareamentos dos grupos bilineares do tipo 1 (OLIVEIRA et al., 2008), que permite a permutabilidade entre termos.

Código 4 – TinyPBC: Declaração dos Eventos

```
1  event encryptMessageA(g).
2  event decryptMessageA(g).
3  event encryptMessageB(g).
4  event decryptMessageB(g).
5
6  event sendIdA(id).
7  event sendIdB(id).
8  event receiveIdA(id).
9  event receiveIdB(id).
10 event mappingOfIDB(g,id).
11 event mappingOfIDA(g,id).
12 event mappingOfIDA2(id).
13 event receivedEncryptedMessageB(bitstring).
14 event encryptMessageB2(g, bitstring).
15 event encryptMessageA2(g, bitstring).
16 event dB(bitstring).
17 event receivedMessageUsingSHKA(bitstring).
18 event receivedMessageUsingSHKB(bitstring).
```

Em Código 4 são declarados os eventos dispostos nos processos

abaixo descritos.

Código 5 – TinyPBC: Nodo A

```
1  let nodeA(ECURVE: g) =
2      event sendIdA(IDA);
3      out(C, IDA);
4      in(C, IDB0: id);
5      event receiveIdB(IDB0);
6      let PB = phi(ECURVE, IDB0) in
7      event mappingOfIDB(ECURVE,IDB0);
8      let KEYA = calcPairKey( calcPrivateKey( PA, S ), PB) in
9      event encryptMessageA(KEYA);
10     new DMSGA:bitstring;
11     event encryptMessageA2(KEYA, DMSGA);
12     out(C, encrypt(KEYA, DMSGA));
13     in(C, EMSGA: bitstring);
14     event receivedMessageUsingSHKA(EMSGA);
15     let (=KEYA, MSG: bitstring) =  decrypt ( EMSGA , KEYA) in
16     event decryptMessageA(KEYA).
```

Em Código 5 o processo que modela o nodo **A** é apresentado. Aqui o nodo inicia enviando seu próprio **id** no canal de comunicação **C**, linha {*3*}, para reconhecimento dos nodos vizinhos. Após, espera-se que nodos vizinhos também enviem seus **ids** no mesmo canal {*4*}.

Caso alguém na rede envie algum **id** (sendo o remetente honestou ou não), o nodo então irá associar o identificador recebido a um ponto na curva elíptica {*6*}. Em {*8*} o nodo então calcula a chave que, pelo **id** recebido, será compartilhada com **B**. O cálculo realizado envolve a chave privada de **A**, representada pelo construtor ***calcPrivateKey(PA, S)***, e a chave pública calculada com o **id** recebido.

Seguindo o que foi descrito nas premissas, estendeu-se o protocolo até um estágio de troca de mensagens simples. Em {*10*} o nodo cria uma mensagem qualquer a ser cifrada e enviada no canal **C**, representado em {*12*}. Após o envio da mensagem cifrada, o nodo então aguarda o recebimento de uma mensagem similarmente construída {*13*}. Verifica-se, utilizando o destrutor responsável pela descriptografia, se a mensagem é composta da mesma chave calculada por **A** e de uma informação qualquer por ela cifrada {*15*}.

Código 6 – TinyPBC: Nodo B

```
1  let nodeB(ECURVE: g) =
2      event sendIdB(IDB);
3      out(C, IDB);
4      in(C, IDA0: id);
5      event receiveIdA(IDA0);
6      let PA = phi(ECURVE, IDA0) in
7      event mappingOfIDA2(IDA0);
8      let KEYB = calcPairKey( calcPrivateKey( PB, S ), PA) in
```

```
9       event encryptMessageB(KEYB);
10      new DMSGB:bitstring;
11      event encryptMessageB2(KEYB, DMSGB);
12      out(C, encrypt(KEYB, DMSGB));
13      in(C, EMSGB: bitstring);
14      event receivedEncryptedMessageB(EMSGB);
15      event receivedMessageUsingSHKB(EMSGB);
16      let (=KEYB, MSG: bitstring) =  decrypt ( EMSGB , KEYB) in
17      event decryptMessageB(KEYB);
18      event dB(MSG).
```

O modelo apresentado em Código 6 é muito similar ao apresentado para o nodo **A**. **B** inicia enviando seu próprio **id** no canal de comunicação **C**, linha {*3*}, para reconhecimento dos nodos vizinhos. Após isso, espera-se que nodos vizinhos também enviem seus **ids** no mesmo canal {*4*}.

Caso alguém na rede envie algum **id** (sendo o remetente honesto ou não), o nodo então irá associar o identificador recebido a um ponto na curva elíptica {*6*}. Em {*8*} o nodo então calcula a chave que, pelo **id** recebido, será compartilhada com **A**. O cálculo realizado envolve a chave privada de **B**, representada pelo construtor ***calcPrivateKey(PB, S)***, e a chave pública calculada com o **id** recebido.

Seguindo o que foi descrito nas premissas, estendeu-se o protocolo até um estágio de troca de mensagens simples. Em {*10*} o nodo cria uma mensagem qualquer a ser cifrada e enviada no canal **C**, representado em {*12*}. Após o envio da mensagem cifrada, o nodo então aguarda o recebimento de uma mensagem similarmente construída {*13*}. Verifica-se, utilizando o destrutor responsável pela descriptografia, se a mensagem é composta da mesma chave calculada por **B** e de uma informação qualquer por ela cifrada {*16*}.

Código 7 – TinyPBC: Inicialização dos Processos

```
1   process
2       let PA = phi(ELIPTICCURVE, IDA) in
3       let PB = phi(ELIPTICCURVE, IDB) in
4           ( !nodeA(ELIPTICCURVE) | !nodeB(ELIPTICCURVE) )
```

Em Código 7 os processos são inicializados paralelamente e replicados ilimitadamente para posterior análise das propriedades de autenticidade.

A análise do protocolo começa então tratando da questão do sigilo. Aqui a informação crucial que se deseja manter fora do alcance dos agentes externos é a chave mestra **S**. Pode-se também estender a análise do sigilo de informações para outros elementos do protocolo que, caso sejam descobertos por agentes externos, não comprometem a

segurança do protocolo.

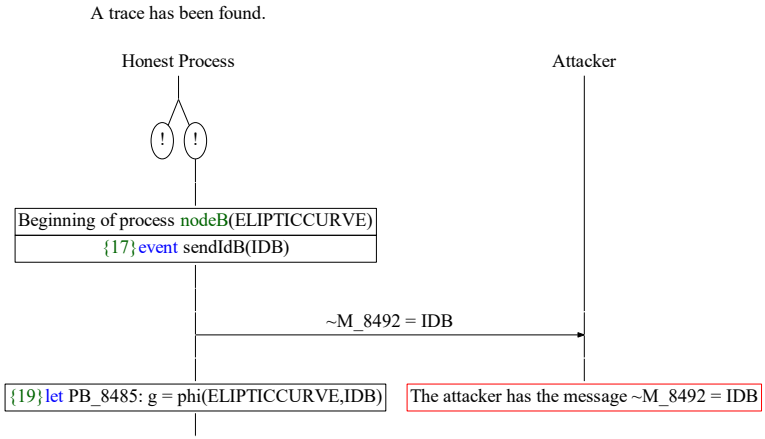
Código 8 – TinyPBC: Consultas de Sigilo

```
query attacker(S).
query attacker(IDA).
query attacker(IDB).
query attacker(ELIPTICCURVE).
```

A análise dos resultados traz como prova a garantia do sigilo da chave mestra **S**. Entretanto, os **ids**, tanto do nodo **A** como do nodo **B** tem seu sigilo violado ao longo do protocolo.

Ambos os **ids** passam a fazer parte do banco de informações de agentes mal intencionados assim que são enviados abertamente na rede para o processo de reconhecimento de nodos vizinhos. As figuras 8 e 9 a seguir ilustram estes comportamentos.

Figura 8 – TinyPBC - Traço Indicativo de Irregularidade: IDb

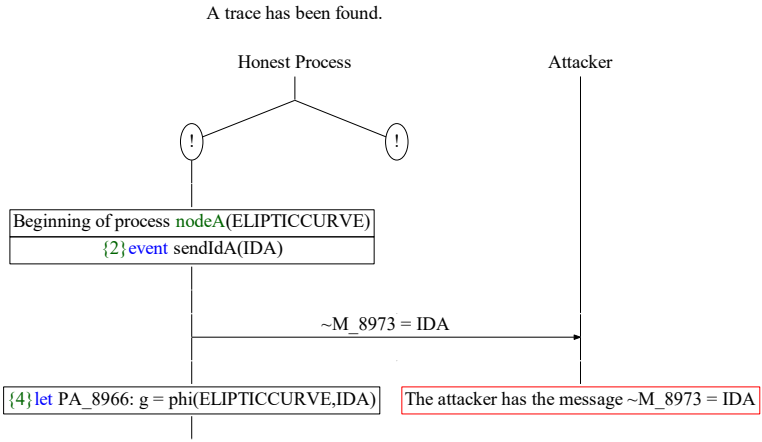


Fonte: do autor.

Além da chave mestra **S**, o protocolo também é capaz de garantir o sigilo da equação que descreve a curva elíptica, denominada arbitrariamente como ***ELIPTICCURVE***.

Como apresentado nas premissas, para se verificar a autenticidade deste protocolo com o ProVerif, estendeu-se o protocolo além do acordo de chaves, incluindo uma simples troca de mensagens entre dois

Figura 9 – TinyPBC - Traço Indicativo de Irregularidade: IDa



Fonte: do autor.

nodos quaisquer que cifram e decifram a mensagem através da chave simétrica que, teoricamente, ambos possuem.

A prova de autenticidade se dá através da distribuição de alguns marcos dentro do fluxo de execução, representados pelos eventos anteriormente declarados, que visam provar a correspondência das mensagens enviadas pelos participantes.

Código 9 – TinyPBC: Consultas de Autenticidade - 1 e 2

```
1  %(*"TPBC-1"*)
2  query SAB: g; event(decryptMessageB(SAB)) ==> event(encryptMessageA(SAB)).
3
4  %(*"TPBC-2"*)
5  query SAB: g; event(decryptMessageB(SAB)) ==> inj-event(encryptMessageA(SAB)).
```

O primeiro teste de autenticidade (identificado por "TPBC-1") visa provar a correspondência entre a criação de uma chave simétrica qualquer aparentemente por **A** e o sucesso que o nodo **B** obtém ao utilizar essa mesma chave para descriptografar uma mensagem por ele recebida. Vale ressaltar que aqui não se verifica nenhum dos parâmetros que compõem a criação da chave simétrica, apenas focando na utilização da mesma no processo de criptografia e descriptografia das

mensagens.

A ferramenta provou que as correspondências acima descritas tem sua garantia mantida pelo protocolo tanto em acordos de nível não injetável, como também em nível de acordo injetável (identificado pelo rótulo "TPBC-2"no código).

Código 10 – TinyPBC: Consultas de Autenticidade - 3 e 4

```
%(*"TPBC-3"*)
query SBA: g; event(decryptMessageA(SBA)) ==> event(encryptMessageB(SBA)).

%(*"TPBC-4"*)
query SBA: g; event(decryptMessageA(SBA)) ==> inj-event(encryptMessageB(SBA)).
```

O terceiro teste (rótulo "TPBC-3"no código), por sua vez, visa provar correspondência entre a criação de uma chave simétrica qualquer aparentemente por **B** e o sucesso que **A** obtém ao utilizar esta chave para descriptografar uma mensagem por ele recebida. Vale ressaltar que aqui não se verifica nenhum dos parâmetros que compõem a criação da chave simétrica, apenas focando na utilização da mesma no processo de criptografia e descriptografia das mensagens.

A ferramenta provou que as correspondências acima descritas mantém tanto a garantia de um acordo não injetável, como também de um acordo injetável ("TPBC-4"no código).

Código 11 – TinyPBC: Consultas de Autenticidade - 5 e 6

```
%(*"TPBC-5"*)
query MSG': bitstring, EC: g, SHKEYA': g, SHKEYB': g, EMSGB': bitstring;
  event(decryptMessageB(SHKEYB')) ==> ( event
    (receivedMessageUsingSHKB(EMSGB')) ==> event(encryptMessageA(SHKEYA'))
      && SHKEYB' = SHKEYA' && EMSGB' = encrypt(SHKEYA', MSG' ) && SHKEYA' =
        calcPairKey( calcPrivateKey(phi(EC, IDA), S), phi(EC, IDB))
          && EC = ELIPTICCURVE).

%(*"TPBC-6"*)
query MSG': bitstring, EC: g, SHKEYA': g, SHKEYB': g, EMSGB': bitstring;
  event(decryptMessageB(SHKEYB')) ==> ( inj-event
    (receivedMessageUsingSHKB(EMSGB')) ==> inj-event(encryptMessageA(SHKEYA'))
      && SHKEYB' = SHKEYA' && EMSGB' = encrypt(SHKEYA', MSG' ) && SHKEYA' =
        calcPairKey( calcPrivateKey(phi(EC, IDA), S), phi(EC, IDB))
          && EC = ELIPTICCURVE).
```

A quinta verificação ("TPBC-5") tem como objetivo aprofundar a prova relacionada a descriptografia da mensagem recebida por **B**. Aqui se verifica se a chave que **B** utilizou para descriptografar o texto recebido ao fim do protocolo é igual a chave simétrica criada pelo nodo **A** como resultado do acordo de chaves entre ambas as entidades. Caso isso seja verdade, significa que de fato foi utilizada a chave privada

do nodo **A** na criação da chave, autenticando a mensagem recebida. Como resultado da análise, o ProVerif conseguiu provar que tal correspondência é garantida tanto em nível não injetável como também em nível injetável ("TPBC-6"no código).

Código 12 – TinyPBC: Consultas de Autenticidade - 7 e 8

```
1  (*"TPBC-7"*)
2  query MSG': bitstring, EC: g, SHKEYA': g, SHKEYB': g, EMSGA': bitstring;
3    event(decryptMessageA(SHKEYA')) ==> ( event
4      (receivedMessageUsingSHKA(EMSGA')) ==> event(encryptMessageB(SHKEYB'))
5        && SHKEYA' = SHKEYB' && EMSGA' = encrypt(SHKEYB', MSG' ) && SHKEYB' =
6          calcPairKey( calcPrivateKey(phi(EC, IDB), S), phi(EC, IDA))
7            && EC = ELIPTICCURVE).
8
9
10 (*"TPBC-8"*)
11 query MSG': bitstring, EC: g, SHKEYA': g, SHKEYB': g, EMSGA': bitstring;
12   event(decryptMessageA(SHKEYA')) ==>
13     ( inj-event (receivedMessageUsingSHKA(EMSGA')) ==>
14         inj-event(encryptMessageB(SHKEYB')) && SHKEYA' = SHKEYB' && EMSGA'
15           = encrypt(SHKEYB', MSG' ) && SHKEYB' =
16             calcPairKey( calcPrivateKey(phi(EC, IDB), S), phi(EC, IDA))
17                 && EC = ELIPTICCURVE).
```

A sétima verificação ("TPBC-7"), por fim, tem como objetivo aprofundar a prova relacionada a descriptografia da mensagem recebida por **A**, similarmente como descrito acima. Aqui se verifica se a chave que **A** utilizou para descriptografar o texto recebido ao fim do protocolo é igual a chave simétrica criada pelo nodo **B** como resultado do acordo de chaves entre ambas as entidades. Caso seja verdade, isso implica que, de fato, foi utilizada a chave privada do nodo **B** na criação da chave, autenticando a mensagem recebida. Como resultado da análise, o ProVerif conseguiu provar que tal correspondência é garantida tanto em nível não injetável como também em nível injetável ("TPBC-8").

6.2.1.1 Sinopse

Nesta seção são resumidos os resultados das verificações do protocolo apresentado por Oliveira et al. (2008). Provou-se que o protocolo é capaz de garantir o sigilo das informações vitais para seu correto funcionamento, mas não de todas as suas informações, assim como mostra a tabela 1.

Já com relação a autenticidade, provou-se inicialmente que existe um acordo em nível injetável da correspondência que descreve a cifragem de uma mensagem - com uma chave simétrica qualquer - e sua descriptografia por parte de **B**, utilizando a mesma chave. A

Tabela 1 – TinyPBC - Sigilo

	S	**GENERATORPOINT**	**IDA**	**IDB**
Sigilo	ok	ok	x	x

mesma relação também foi provada com relação a chave utilizada na descriptografia que **A** performa.

Ao aprofundar as consultas acima, ficou provado que, também em nível injetável, existe uma correspondência entre a utilização a utilização da chave privada de **A** para a criação da chave simétrica utilizada para cifrar a mensagem recebida por **B**, e sua consequente descriptografia com a chave simétrica calculada por **B**.

Por fim, ficou também provado que, em nível injetável, existe uma correspondência entre a utilização a utilização da chave privada de **B** para a criação da chave simétrica utilizada para cifrar a mensagem recebida por **A**, e sua consequente descriptografia com a chave simétrica calculada por **A**. A tabela 2 organiza as informações acima apresentadas.

Tabela 2 – TinyPBC - Autenticidade

TINYPBC-	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Não Injetável	ok		ok		ok		ok	
Injetável		ok		ok		ok		ok

6.2.2 Herrera

A análise do protocolo apresentado por Herrera e Hu se baseia integralmente no trabalho de Herrera e Hu (2012), escrito pelos desenvolvedores e idealizadores do protocolo em questão. Algumas premissas foram estabelecidas para a adequação do protocolo ao modelo proposto a ser analisado pelo ProVerif. São elas:

- A primeira premissa traz que agentes externos não possuem acesso físico aos nodos e estações base, sendo impossível a adulteração ou captura física aos dispositivos;
- A segunda premissa trata da questão da recenticidade das informações, que, com base na análise do texto que descreve o protocolo, não traz nenhuma indicação de tempo na troca de mensagens entre os nodos. Não podendo assim se levar em conta a questão da recenticidade na análise da autenticidade;
- A terceira premissa adotada traz que o protocolo já finalizou a fase de pré distribuição dos nodos e tem seu início em estado de pós distribuição, já em ambiente hostil;
- A quarta premissa diz que informações atribuídas aos nodos em fase de pré distribuição não estão inicialmente disponíveis aos agentes mal intencionados. Em outras palavras, apesar de os dados não estarem no banco de informações inicial dos agentes externos, isso não impede que, durante o fluxo de execução do protocolo, estas informações não sejam inferidas pelo / enviadas para agentes externos.
- A quinta premissa está relacionada ao mecanismo de defesa contra ataques de repetição que é implementado no protocolo. Como o autor não especifica qual o tratamento dado aos *nonces* aferidos como repetidos, assume-se então que:
    1. a cargo de verificação, cada *nonce* é então associado ao nodo que o enviou e, tal relação fica armazenada na estação base;
    2. relações de nodo/*nonce* repetidas recebidas pela estação base causam a interrupção da continuidade do protocolo para aquela seção.

Partindo das premissas acima, modelou-se o protocolo em questão em *Applied Pi Calculus*. O código do fonte do modelo é destrinchado e explicado abaixo. O cenário adotado para as verificações trata

a troca de mensagens entre um nodo qualquer denominado arbitrariamente como **S1** e uma estação base também denominada arbitrariamente como **BS**.

Aqui, apresenta-se o cenário onde o nodo **S1** deseja obter uma chave simétrica para comunicações futuras com a estação base **BS**.

Código 13 – Herrera: Tipos de Dados

```
1  type skey.
2  type nonce.
3  type id.
4  type pkey.
5  type shkey.
6  type ackmsg.
```

Em Código 13 definem-se os tipos de dados a serem tratados durante o fluxo de execução do protocolo. O tipo **skey** é utilizado para representar as chaves privadas de cada participante. Já **nonce**, simboliza os *nonces* utilizados a serem criados pelos participantes. Os identificadores dos participantes da rede são denotados pelo tipo **id**. As chaves públicas são representadas por **pkey**, e a chave simétrica criada pela estação base é do tipo **shkey**. Por fim, o último tipo trata da mensagem de *acknowledgement* final enviada pelo nodo.

Código 14 – Herrera: Nomes Livres e Constantes

```
1  free BSSKEY: skey [private].
2  free C: channel.
3  free S1SKEY: skey [private].
4  free S1: id [private].
5  free ACK:ackmsg.
6  table nonces(id, nonce).
```

Já em Código 14, trata-se da declaração dos nomes livres (*free names*). A chave privada da estação base é chamada de **BSSKEY**, enquanto a chave privada do nodo, que tem seu identificador definido como **S1**, é chamada de **S1SKEY**. Assim como na seção anterior, **C** é o nome dado ao canal de comunicação para a troca de mensagens. **ACK** é a mensagem criada ao fim do protocolo pelo nodo **S1** para finalizar a comunicação. Por fim, **table nonces()** é a base de dados que armazena a relação entre os **nonces** enviados e os nodos que os enviaram, os relacionando através do identificador.

Código 15 – Herrera: Construtores e Destrutores

```
1  fun pk(skey): pkey.
2
3  fun aenc(shkey, pkey): bitstring.
4  reduc forall SHAREDKEY: shkey, NODEPKEY: skey;
5   adec(aenc(SHAREDKEY, pk(NODEPKEY)), NODEPKEY) = SHAREDKEY.
6
7  fun calcSharedKey(nonce, id): shkey.
8
9  fun hash(shkey): bitstring.
10
11 fun sign(bitstring, skey): bitstring.
12 reduc forall HASHVALUE: bitstring, BASESKEY: skey;
13   checkSign(sign(HASHVALUE, BASESKEY), pk(BASESKEY)) = HASHVALUE.
```

Os construtores e destrutores que manipulam os dados do fluxo de execução do protocolo são apresentados em Código 15. O primeiro construtor, presente na linha {*1*}, modela a criação de chaves públicas usando como parâmetro chaves privadas. O segundo construtor, descrito na linha {*3*} modela a criptografia assimétrica de uma chave simétrica por uma chave pública, resultando em uma *string* de *bits*. O destrutor descrito nas linhas {*4*} e {*5*} modela o processo complementar da criptografia assimétrica, onde os dados cifrados são descriptografados pela chave privada exigida.

Na linha {*7*} é representada a criação da chave simétrica com base no conjunto formado por um ***nonce*** e um **id**. Já na linha {*9*} se descreve o construtor que simboliza a criação do valor *hash* de uma chave simétrica, gerando também uma *string* de *bits*. Por fim, o processo de assinatura digital é modelado na linha {*11*}, onde uma *string* de *bits* é cifrada com uma chave privada. O construtor a ele atrelado é descrito nas linhas {*12*} e {*13*}, onde o valor acima mencionado é então descriptografado utilizando a chave pública que complementa o processo.

Código 16 – Herrera: Tipos de Dados

```
1  event createSharedKey(nonce,id).
2  event sendIDandNonce(id,nonce).
3  event successfullyCheckedTheSign(bitstring).
4  event createTheSharedKeyHash(bitstring).
5  event encryptedSharedKeyReceived(bitstring).
6  event signedMessageReceived(bitstring).
7  event encryptionOfTheSharedKey(bitstring).
8  event decryptSharedKey(shkey).
9  event nonceStoredInTheBaseStation(id, nonce).
10 event nonceAlreadyUsed(id, nonce).
11 event idNotFound(id).
12 event afterTheSharedKeyCreation(shkey).
13 event createTheSignedMessage(bitstring).
14 event idAndNonceReceived(id, nonce).
15 event decryptSharedKey2(shkey,skey).
16 event s1PublicKeyCreation(skey).
```

```
17  event t2(shkey, pkey).
18  event afterUnsigningHash(bitstring).
```

Os eventos dispostos ao longo da execução do protocolo são então declarados em Código 16.

Código 17 – Herrera: Nodo S1

```
1   let nodeS1(S1:id) =
2       new N1:nonce;
3       event sendIDandNonce(S1,N1);
4       out(C, (S1, N1));
5       in(C, (ENCRYPTEDSHAREDKEY:bitstring, SIGNEDMESSAGE:bitstring));
6       event encryptedSharedKeyReceived(ENCRYPTEDSHAREDKEY);
7       event signedMessageReceived(SIGNEDMESSAGE);
8       let KS1BS = adec(ENCRYPTEDSHAREDKEY, S1SKEY) in
9       event decryptSharedKey(KS1BS);
10      event decryptSharedKey2(KS1BS, S1SKEY);
11      let HASHS1 = hash(KS1BS) in
12      let UNSIGNEDHASH = checkSign(SIGNEDMESSAGE, pk(BSSKEY)) in
13      event afterUnsigningHash(UNSIGNEDHASH);
14      if (HASHS1 = UNSIGNEDHASH) then
15      event successfullyCheckedTheSign(UNSIGNEDHASH);
16      out(C, ACK).
```

Em Código 17 o comportamento do nodo **S1** é modelado. O nodo começa em {*2*} gerando um ***nonce*** que irá ser enviado juntamente com seu próprio **id** no canal de comunicação **C** {*4*}, representando assim um requerimento para uma chave simétrica para a estação base. Após o envio do requerimento da chave, o nodo então aguarda o recebimento de uma mensagem que concatena duas *strings* de *bits* {*5*}. **S1** então tenta descriptografar o primeiro dos dados concatenados utilizando sua própria chave privada {*8*}. Em caso de sucesso, **S1** continua seu fluxo de execução e obtém um valor *hash* da chave que acabou de descriptografar {*11*}.

Para verificar a assinatura digital da estação base, **S1** então tenta descriptografar a segunda parte da mensagem recebida utilizando a chave pública da estação base {*12*}. Em caso de sucesso, este valor é comparado com o valor obtido anteriormente na linha {*11*}. Se ambos os valores forem iguais {*14*}, significa que, se o sigilo da chave privada da estação base foi mantido, então de fato a chave recebida foi enviada pela estação base e seu valor não foi alterado no processo. O nodo **S1** então finaliza seu processo enviando uma mensagem de *acknowledgment* no canal **C** {*16*}.

Código 18 – Herrera: Estação Base

```
let baseStation() =
        in(C, (SX:id, NX: nonce));
        event idAndNonceReceived(SX, NX);
        get nonces(=SX, N': nonce) in(
        get nonces(=SX, =NX) in (
                event nonceAlreadyUsed(SX, NX)
        )
        else
                insert nonces(SX, NX);
                event nonceStoredInTheBaseStation(SX, NX);
                event createSharedKey(NX,SX);
                let KBSS1 = calcSharedKey(NX, SX) in
        event afterTheSharedKeyCreation(KBSS1);
        event s1PublicKeyCreation(S1SKEY);
        let PS1SKEY = pk(S1SKEY) in
        let MSGKEY = aenc(KBSS1,PS1SKEY) in
        event t2(KBSS1, PS1SKEY);
        event encryptionOfTheSharedKey(MSGKEY);
        let MSGHASH = hash(KBSS1) in
        event createTheSharedKeyHash(MSGHASH);
        let MSGSIGNED = sign(MSGHASH, BSSKEY) in
        event createTheSignedMessage(MSGSIGNED);
        out(C, (MSGKEY, MSGSIGNED));
        in(C, ACKN:ackmsg)
        )
        else
                event idNotFound(SX).
```

O fluxo de execução da estação base, por sua vez, é modelado conforme o Código 18. O processo inicia quando ocorre o recebimento de um requerimento de chave simétrica, composto de um **id** e de um ***nonce*** {*2*}. A estação base então busca se o **id** fornecido se encontra em sua base de dados {*4*}. Em caso negativo, o processo é abortado, comportamento modelado em {*26*} e {*27*}. Caso contrário, a estação base busca se o ***nonce*** fornecido já foi associado ao **id** do nodo em questão {*5*}, sendo que um resultado positivo dessa busca também resulta na interrupção do fluxo de execução. Entretanto, no caso onde o ***nonce*** ainda não havia sido utilizado {*8*} , a estação base armazena este valor em sua base de dados {*9*} e a partir dele gera a chave simétrica a ser enviada para o nodo ao qual o **id** representa {*12*}.

A etapa de cifragem da chave simétrica acontece em {*16*}, onde o construtor de criptografia assimétrica cifra a chave **KBSS1** utilizando a chave pública de **S1**. O valor *hash* da chave simétrica é obtido em {*19*}, sendo ele assinado digitalmente em {*21*}, onde o mesmo é cifrado utilizando a chave privada da estação base. A estação base então concatena a chave simétrica criptografada e o valor *hash* assinado e os envia no canal de comunicação **C** {*23*}. Por fim, a estação base aguarda uma confirmação de que o processo foi bem sucedido por parte de **S1** {*14*}.

Código 19 – Herrera: Inicialização dos Processos

```
process
        new NPH: nonce;
        insert nonces(S1,NPH);
        ( !nodeS1(S1) | !baseStation() )
```

A inicialização tanto do nodo **S1**, como também da estação base é descrita no Código 19, onde ambos os processos são executados paralelamente em quantidades ilimitada de instancias dos dois $\{4\}$. Em $\{3\}$, adiciona-se o **id** referente ao nodo **S1** na base de dados juntamente com um ***nonce*** qualquer objetivando garantir que quando **S1** enviasse um requerimento para a estação, que o mesmo não fosse negado.

Código 20 – Herrera: Consultas de Sigilo

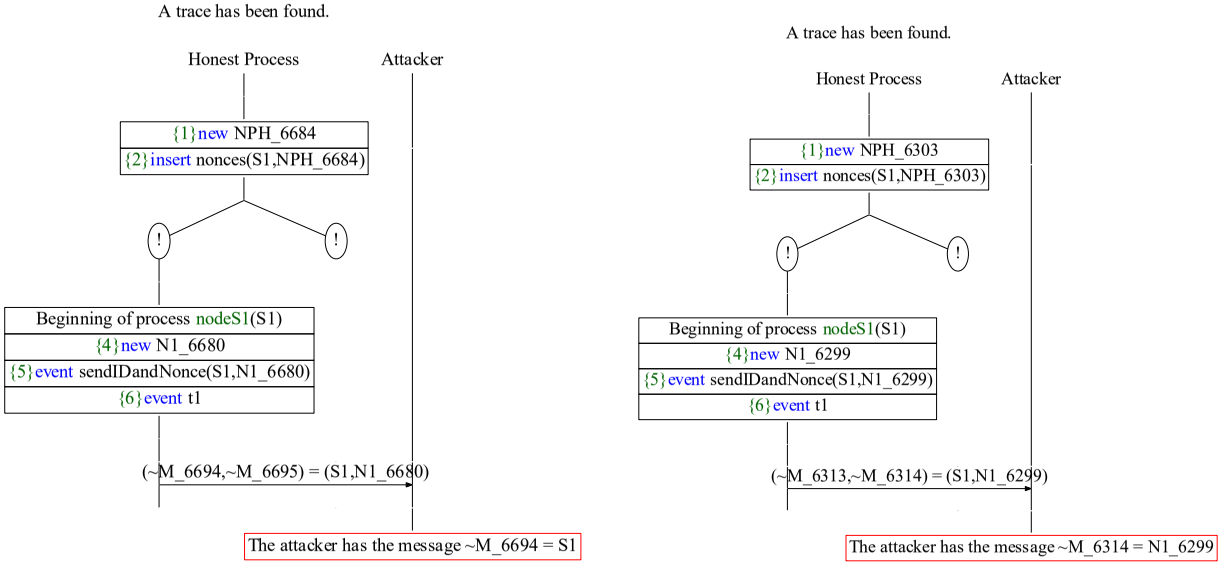
```
query attacker(BSSKEY).
query attacker(S1SKEY).
query attacker(S1).
query attacker(new N1).
```

A análise do protocolo começa então tratando da questão do sigilo, onde o protocolo se utiliza de duas informações essenciais para garantir a segurança da troca de informações. A primeira que se deseja verificar o sigilo é a chave privada **S1SKEY**. A análise mostra que o protocolo consegue garantir a manutenção do sigilo de **S1SKEY**.

A segunda meta a ser alcançada é a manutenção da propriedade de sigilo da chave privada da estação base **BSSKEY** na codificação do modelo. O resultado da verificação traz que o protocolo é capaz de manter o sigilo de **BSSKEY** durante toda a sua execução.

Verificou-se ainda o sigilo de duas outras informações, o **ID** do nodo **S1** e o ***nonce*** por ele gerado. Como ambos necessitam ser enviados abertamente (leia-se sem cifragem) para realizar a requisição de chave para a estação base **BS**, este valores passam a fazer parte do conjunto de informações que os agentes externos tem conhecimento. A figuras 10 ilustra os traços encontrados no **id** no ***nonce*** utilizados por **S1**.

Figura 10 – Herrera - Traço Indicativo de Irregularidade: id e *nonce*.



Fonte: do autor.

A prova de autenticidade se dá, assim como na verificação do TinyPBC, através da distribuição de eventos dentro do código que visam provar a correspondência das mensagens enviadas pelos participantes. Os principais eventos que comprovam a garantia de autenticidade do protocolo são abaixo descritos.

Código 21 – Herrera: Consultas de Autenticidade - 1 e 2

```
1  %(*HERRERA-1*)
2  query SHK': shkey; event(decryptSharedKey(SHK'))
3      ==> event(afterTheSharedKeyCreation(SHK')).
4
5  %(*HERRERA-2*)
6  query SHK': shkey; event(decryptSharedKey(SHK'))
7      ==> inj-event(afterTheSharedKeyCreation(SHK')).
```

A primeira das verificações (identificada pelo rótulo "HERRERA-1"no código) trata da prova de correspondência entre a descriptografia de uma chave cifrada recebida pelo nodo **S1** e se existiu um processo no fluxo do protocolo previsto a acontecer que a deu origem. De fato a ferramenta consegue provar que existiu uma etapa anterior a descriptografia da chave recebida que envolveu a cifragem de tal chave.

Entretanto, esta verificação não é suficiente para provar que, de fato, a mensagem partiu da estação base, visto que a cifragem da chave compartilhada em questão foi realizada com a chave pública de **S1**, sendo, portanto, acessível por qualquer participante. Além disso, a injetividade de tal correspondência não é válida, como mostra a consulta identificada pelo rótulo "HERRERA-2"no código.

Código 22 – Herrera: Consultas de Autenticidade - 3

```
1  %(*HERRERA-3*)
2  query S': id, N': nonce; event(createSharedKey(N',S'))
3      ==> event(sendIDandNonce(S',N')).
```

A consulta identificada como "HERRERA-3"trata da criação da chave compartilhada entre a base e o nodo **S1**, executada pela estação base após o envio de um **id** e um ***nonce*** pelo nodo **S1**. A verificação de tal correspondência traz que a autenticação entre os nodos não é válida, visto que tanto o **id** como o ***nonce*** são enviados de forma desprotegida na rede, passando assim a fazer parte da base de conhecimento de agentes externos.

Código 23 – Herrera: Consultas de Autenticidade - 4 e 5

```
%(*HERRERA-4*)
query SHK': shkey, PKEY1: pkey, SKEY1: skey, MT: bitstring;
  event(decryptSharedKey2(SHK', SKEY1)) ==>
    ( event(encryptionOfTheSharedKey(MT))
     ==> event(afterTheSharedKeyCreation(SHK'))&&
         MT = aenc(SHK',PKEY1) && PKEY1 = pk(SKEY1)).

%(*HERRERA-5*)
query SHK': shkey, PKEY1: pkey, SKEY1: skey, MT: bitstring;
  event(decryptSharedKey2(SHK', SKEY1)) ==>
    ( inj-event(encryptionOfTheSharedKey(MT))
    ==> inj-event(afterTheSharedKeyCreation(SHK'))&&
         MT = aenc(SHK',PKEY1) && PKEY1 = pk(SKEY1)).
```

A consulta de correspondência entre a chave descriptografada por **S1** e as etapas performadas pela estação base para sua criação e cifragem desta chave é descrita em "HERRERA-4", onde se visa verificar se a fase de descriptografia assimétrica da chave recebida, que utilizando a chave privada de **S1**, foi em decorrência de eventos passados que descrevem a criação da chave em questão e sua cifragem através da utilização da chave pública de **S1**.

O resultado da verificação traz que o protocolo consegue garantir tal correspondência em nível não injetável. Ao tentar provar a correspondência acima em nível injetável (rótulo "HERRERA-5"), a ferramenta não foi capaz de fornecer uma comprovação de garantia, assim como também não foi capaz de apresentar um traço de ataque que invalidasse a correspondência.

Código 24 – Herrera: Consultas de Autenticidade - 6 e 7

```
%(*HERRERA-6*)
query HASHBS: bitstring, HASHS1: bitstring, MSIGHASH: bitstring,
  S1': id, N': nonce, MSGKEY': bitstring, KBSS1': shkey;
        event(afterUnsigningHash(HASHS1))        ==>
          ( event(createTheSignedMessage(MSIGHASH))
                ==> ( event(afterTheSharedKeyCreation(KBSS1'))
                        ==> event(nonceStoredInTheBaseStation(S1', N'))))
                                && HASHS1 = HASHBS && MSIGHASH =
                                  sign(HASHBS, BSSKEY)
                                    && HASHBS = hash(KBSS1')
                                      && KBSS1' = calcSharedKey(N', S1').

%(*HERRERA-7*)
query HASHBS: bitstring, HASHS1: bitstring, MSIGHASH: bitstring,
  S1': id, N': nonce, MSGKEY': bitstring, KBSS1': shkey;
        inj-event(afterUnsigningHash(HASHS1)) ==>
          ( inj-event(createTheSignedMessage(MSIGHASH))
                ==> ( inj-event(afterTheSharedKeyCreation(KBSS1'))
                        ==> inj-event(nonceStoredInTheBaseStation(S1', N'))))
                                && HASHS1 = HASHBS && MSIGHASH =
                                  sign(HASHBS, BSSKEY)
                                    && HASHBS = hash(KBSS1')
                                      && KBSS1' = calcSharedKey(N', S1').
```

Por fim, a questão da garantia de autenticidade da mensagem assinada digitalmente é um dos pontos chave na verificação do protocolo. Na consulta "HERRERA-6", verifica-se se o valor de *hash* obtido através da chave descriptografada é igual ao valor de *hash* utilizado na assinatura digital e se a mensagem foi de fato assinada com a chave privada da estação base.

Verifica-se também se a criação da chave utilizada na obtenção do valor de *hash* foi em decorrência de valores armazenados na base de dados da estação base que controla os ***nonces*** associados aos **ids** dos nodos. Como resultado da análise, a ferramenta apresentou que o protocolo é capaz de garantir a correspondência acima descrita em nível não injetável. Entretanto, a ferramenta mais uma vez não foi capaz de fornecer uma definição quanto a garantia ou não de tal propriedade em nível injetável.

6.2.2.1 Sinopse

Resumindo os resultados, provou-se que o protocolo é capaz de garantir o sigilo das informações vitais para seu correto funcionamento, mas não de todas as suas informações, assim como mostra a tabela 3.

Tabela 3 – Herrera - Sigilo

	S1SKEY	**BSSKEY**	**S1**	**N1**
Sigilo	ok	ok	x	x

Já com relação à autenticidade, ficou provado que o protocolo é capaz de garantir um acordo em nível não injetável da correspondência que descreve a criação de uma chave e sua seguida descriptografia por **S1**, mas sem informações sobre quem a criou.

Durante as verificações, provou-se também que a correspondência entre a criação da chave simétrica pela estação base e os dados enviados a ela enviados não é válida, visto que o **id** e o **nonce** são abertamente enviados na rede.

Com relação a descriptografia dos dados recebidos por **S1**, provou-se que existe, em nível de acordo não injetável, uma correspondência entre a cifragem da chave simétrica utilizando a chave pública de **S1** e sua posterior descriptografia por **S1**.

Por fim, verificou-se a correspondência entre a assinatura digital realizada pela estação base e a sua validação pelo nodo **S1**. Provou-se que esta relação é válida em nível de acordo não injetável, mas não se

foi capaz de chegar à uma conclusão sobre o nível injetável. A tabela 4 organiza os resultados das consultas previamente apresentadas, onde **ok** representa a garantia da consulta, **x** a não validade da consulta e **?** a não capacidade da ferramenta em definir o resultado.

Tabela 4 – Herrera - Autenticidade

HERRERA-	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Não Injetável	ok		x	ok		ok	
Injetável		x			?		?

6.2.3 Najmus Saqib

A seguir serão apresentados resultados da verificação formal do protocolo proposto por Najmus Saqib. A análise do protocolo em questão se baseia integralmente no texto apresentado por Saqib (2016), escrito pelo próprio desenvolvedor e idealizador do protocolo. Algumas premissas foram estabelecidas para a adequação do protocolo ao modelo proposto a ser analisado pelo ProVerif. São elas:

- A primeira premissa traz que agentes externos não possuem acesso físico aos nodos e estações base, sendo impossível a adulteração ou captura física aos dispositivos;
- A segunda premissa trata da questão da recenticidade das informações, que, com base na análise do texto que descreve o protocolo, não traz nenhuma indicação de tempo na troca de mensagens entre os nodos. Não podendo assim se levar em conta a questão da recenticidade na análise da autenticidade;
- A terceira premissa adotada traz que o protocolo já finalizou a fase de pré distribuição dos nodos e tem seu início em estado de pós distribuição, já em ambiente hostil;
- A quarta premissa diz que informações atribuídas aos nodos em fase de pré distribuição não estão inicialmente disponíveis aos agentes mal intencionados. Em outras palavras, apesar de os dados não estarem no banco de informações inicial dos agentes externos, isso não impede que, durante o fluxo de execução do protocolo, estas informações não sejam inferidas pelo / enviadas para agentes externos.

Partindo das premissas acima, modelou-se o protocolo em questão em *Applied Pi Calculus*. O código do fonte do modelo é destrinchado e explicado abaixo. O cenário modelado para as verificações trata a troca de mensagens entre dois nodos quaisquer denominados arbitrariamente como **A** e **B**.

Como já explicado nas premissas, este cenário se passa após a distribuição dos nodos em ambiente hostil, o que faz com que o servidor base não tenha participação na troca de mensagens entre os nodos. Aqui, inicialmente, se é apresentado um cenário onde o nodo **A** deseja se comunicar com o nodo **B**.

Código 25 – Najmus Saqib: Tipos de Dados

```
type skey.
type g.
type number.
type secretkey.
type pkey.
```

Em Código 25 são definidos os tipos de dados utilizados na modelagem do protocolo. As chaves privadas aqui são representadas pelo termo **skey**, enquanto as chaves públicas são referenciadas através do termo **pkey**. Os valores relativos aos dados que envolvem a equação da curva elíptica e os pontos a ela relacionados são categorizados como do tipo **g**. Números aleatórios gerados ao longo do protocolo são tratados como **numbers**. Por fim, o tipo do valor que é utilizado na criação da chave simétrica é chamado de **secretkey**.

Código 26 – Najmus Saqib: Nomes Livres e Constantes

```
free C:channel.
const GENERATORPOINT: g [private].
free KA: skey [private].
free KB: skey [private].
```

Acima são tratadas as definições dos nomes livres (*free names*). O canal de comunicação é, como nos demais protocolos analisados, chamado de **C**. **GENERATORPOINT** é o nome atribuído ao ponto inicial da da curva elíptica utilizada nos processos de cifragem do protocolo. As chaves privadas dos nodos **A** e **B** são denominadas, respectivamente, como **KA** e **KB**.

Código 27 – Najmus Saqib: Construtores e Destrutores

```
fun calcSecretKey(number, g): secretkey.

fun calcPublicKey(skey, g): pkey.

fun addsSkeyPlusNumber(skey, number): g.

fun encryption(g,pkey): g.

reduc forall KA': skey, X': number, GENPOINT: g,
    KB': skey, PUA': pkey; decryption
        ( encryption( addsSkeyPlusNumber(KA',X'),
            calcPublicKey(KB', GENPOINT)), KB',
                calcPublicKey(KA', GENPOINT) )
                    = calcSecretKey(X', GENPOINT).

fun calcSharedKey(secretkey, secretkey): secretkey.
```

Em Código 27 são modelados os construtores e os destrutores do protocolo. O primeiro construtor, linha {*1*}, tem como objetivo receber um número qualquer e um ponto na curva elíptica e gerar uma

chave secreta. O segundo construtor, por sua vez, objetiva calcular a chave pública de um participante qualquer {*3*}. O construtor presente em {*5*} é utilizado para realizar a soma da chave privada de um participante a ao número aleatório por ele gerado. A cifragem dos valores resultantes das somas performadas pelo construtor em {*5*} é executada pelo construtor descrito em {*7*}.

O destrutor associado a cifragem acima descrita, representado entre as linhas {*9*} e {*14*}, modela que o desfecho da descriptografia traz como resultado a chave secreta calculada do nodo que, em teoria, enviou a mensagem. Por fim, o construtor em {*16*} modela que o cálculo da chave simétrica a ser compartilhada entre os nodos é resultado da soma de duas chaves secretas que produzem um resultado também do tipo chave secreta.

Código 28 – Najmus Saqib: Declaração dos Eventos

```
1  event nodeACreatesTheSecretKey(secretkey).
2  event nodeBCreatesTheSecretKey(secretkey).
3
4  event nodeBDecryptsAsSecretKey(secretkey).
5  event nodeADecryptsBsSecretKey(secretkey).
6
7  event nodeBComputesSharedKey(secretkey).
8  event nodeAComputesSharedKey(secretkey).
9
10 event sendKAXPUB(g).
11 event sendKBYPUA(g).
12
13 event receiveEncryptedMsgB(g).
14 event receiveEncryptedMsgA(g).
15
16 event decryptedUsingKB(secretkey,skey).
17 event decryptedUsingKA(secretkey,skey).
```

Os eventos dispostos ao longo da execução do protocolo são então declarados em Código 28.

Código 29 – Najmus Saqib: Nodo A

```
let nodeA(PUB: pkey) =
    new X: number;
    let XG = calcSecretKey(X,GENERATORPOINT) in
    event nodeACreatesTheSecretKey(XG);
    let KAXPUB = encryption(addsSkeyPlusNumber(KA, X), PUB) in
    event sendKAXPUB(KAXPUB);
    out(C, KAXPUB);
    in(C, K2:g);
    event receiveEncryptedMsgA(K2);
    let GY = decryption(K2, KA, PUB) in
    event decryptedUsingKA(GY,KA);
    event nodeADecryptsBsSecretKey(GY);
    let XGGY = calcSharedKey(XG, GY) in
```

```
14      event nodeAComputesSharedKey(XGGY).
```

Em Código 29 o comportamento do nodo **A** é modelado, iniciando sua execução ao gerar o número aleatório **X**, como apresentado na linha {*2*}. Dando continuidade, **A** então calcula sua chave secreta **XG**, que é resultado da multiplicação de **X** e do ponto inicial da curva elíptica **GENERATORPOINT** {*3*}. O valor de **X** é somado a chave privada **KA** e cifrado pela chave pública **KB** {*5*} para ser enviado posteriormente no canal **C** {*7*}. **A** aguarda, no canal **C**, então por uma chave similarmente construída {*8*}. **A** em {*10*} descriptografa então a chave **K2** recebida em {*8*}, trazendo como resultado, se bem sucedido, a chave secreta do nodo **B**. Por fim, **A** calcula a chave simétrica, resultado da soma da sua própria chave secreta **XG** e a chave recém descriptografada de **B**, **GY** {*13*}.

Código 30 – Najmus Saqib: Nodo B

```
1   let nodeB(PUA: pkey) =
2       new Y: number;
3       let YG = calcSecretKey(Y, GENERATORPOINT) in
4       event nodeBCreatesTheSecretKey(YG);
5       in(C, K1: g);
6       event receiveEncryptedMsgB(K1);
7       let GX = decryption(K1,KB,PUA) in
8       event decryptedUsingKB(GX,KB);
9       event nodeBDecryptsAsSecretKey(GX);
10      let YGGX = calcSharedKey(GX,YG) in
11      event nodeBComputesSharedKey(YGGX);
12      let KBYPUA = encryption(addsSkeyPlusNumber(KB,Y),PUA) in
13      event sendKBYPUA(KBYPUA);
14      out(C, KBYPUA).
```

Já em Código 30, apresenta-se o fluxo de execução no nodo **B**. Assim como em **A**, **B** inicia seu processo escolhendo um número aleatório , aqui chamado de **Y** {*2*} e, a partir dele, gerar sua chave secreta **YG** em {*4*}. A diferença entre a execução dos nodos começa em {*5*}, onde **B**, por não ser o participante a iniciar a comunicação, deve esperar uma chave a ser enviada no canal **C** {*5*}. Após o recebimento da chave em questão, **B** tenta a descriptografar utilizando sua própria chave privada **KB** e a chave pública de **A**, **PUA** {*7*}. Caso o processo seja bem sucedido, **B** calcula então a chave simétrica a ser utilizada com **A** ao somar sua própria chave secreta **YG** com a chave secreta recém descriptografada, **GX** {*10*}. Para finalizar o processo de acordo de chaves, **B** então cifra **Y** utilizando sua própria chave privada **KB** e a chave pública de **A**, **PUA** {*12*} e envia o resultado no canal de comunicação aberto **C** {*14*}.

Código 31 – Najmus Saqib: Inicialização dos Processos

```
process
        let PUA = calcPublicKey(KA, GENERATORPOINT) in
                let PUB = calcPublicKey(KB, GENERATORPOINT) in
                        out(C, PUA);
                        out(C, PUB);
                        ( !nodeA(PUB) | !nodeB(PUA) )
```

Tanto o processo referente ao nodo **A**, como também o do nodo **B** são inicializados paralelamente em um número ilimitado de instancias, como apresentado em Código 31.

Código 32 – Najmus Saqib: Consultas de Sigilo

```
query attacker(KA).
query attacker(KB).
query attacker(new X).
query attacker(new Y).
query attacker(GENERATORPOINT).
```

Com relação ao sigilo, o protocolo se utiliza de quatro informações essenciais para garantir a segurança da troca de informações. A primeira que se deseja verificar a garantia do sigilo é a chave privada **KA**. O resultado da verificação formal traz que o protocolo consegue garantir o sigilo da chave **KA** durante toda sua execução.

A segunda meta a ser alcançada é o sigilo da chave privada **KB**. O resultado da verificação formal traz que o protocolo consegue também garantir o sigilo da chave **KB** durante toda sua execução.

A terceira e quarta meta a serem alcançadas são os sigilos dos valores aleatórios escolhidos arbitrariamente pelos nodos **A** e **B**, chamados arbitrariamente de **X** e **Y**, respectivamente. Assim como as chaves privadas, os valores aleatórios também conseguem ser mentidos em sigilo pelo protocolo ao longo de todo seu fluxo de execução.

Como verificação extra de sigilo, analisou-se a garantia de sigilo da equação que descreve o ponto inicial presente curva elíptica. O resultado trouxe que o protocolo também consegue garantir a manutenção do sigilo do ponto.

A prova de autenticidade se dá, assim como na verificação dos dois protocolos anteriores, através da distribuição de eventos dentro do código que visam provar a correspondência das mensagens enviadas pelos participantes. Os principais eventos que avaliam a garantia de autenticidade do protocolo estão descritos abaixo.

Código 33 – Najmus Saqib: Consultas de Autenticidade - 1 e 2

```
%(*"NAJMUS-1"*)
query YG': secretkey, GX': secretkey, YGXG': secretkey, X':number;
    event(nodeBComputesSharedKey(YGXG'))
        ==> event(nodeACreatesTheSecretKey(GX'))
            && YGXG' = calcSharedKey(GX', YG')
                && GX' = calcSecretKey(X',GENERATORPOINT).

%(*"NAJMUS-2"*)
query YG': secretkey, GX': secretkey, YGXG': secretkey, X':number;
    event(nodeBComputesSharedKey(YGXG'))
            ==> inj-event(nodeACreatesTheSecretKey(GX'))
                && YGXG' = calcSharedKey(GX', YG')
                    && GX' = calcSecretKey(X',GENERATORPOINT).
```

A primeira das verificações (rótulo "NAJMUS-1") trata de analisar se a chave secreta criptografada recebida pelo nodo **B** que tem sua descriptografia dada pela

- chave privada de **B**, **KB**- inferindo assim, que a mesma foi cifrada utilizando chave pública de **B**, **PUB**;
- e pela chave pública de **A**, **PUA** - inferindo que a chave privada de **A**, **KA** - também fez parte do seu processo de criação

partiu, de fato, do nodo que gerou o valor aleatório que resulta na chave secreta descriptografada por **B**. O ProVerif prova que o protocolo é capaz sim de garantir a correspondência entre os eventos e o conjunto de informações por eles manipulado em nível não injetável. Entretanto, através da consulta de nível injetável (rótulo "NAJMUS-2"), a ferramenta trouxe como inválida tal correspondência.

Código 34 – Najmus Saqib: Consultas de Autenticidade - 3 e 4

```
%(*"NAJMUS-3"*)
query YG': secretkey, GX': secretkey, YGXG': secretkey, Y':number;
    event(nodeAComputesSharedKey(YGXG'))
                ==> event(nodeBCreatesTheSecretKey(YG'))
                    && YGXG' = calcSharedKey(GX', YG')
                        && YG' = calcSecretKey(Y',GENERATORPOINT).
%(*"NAJMUS-4"*)
query YG': secretkey, GX': secretkey, YGXG': secretkey, Y':number;
    event(nodeAComputesSharedKey(YGXG'))
            ==> inj-event(nodeBCreatesTheSecretKey(YG'))
                && YGXG' = calcSharedKey(GX', YG')
                    && YG' = calcSecretKey(Y',GENERATORPOINT).
```

A terceira verificação (rótulo "NAJMUS-3") trata de analisar se a chave secreta criptografada recebida pelo nodo **A** que tem sua descriptografia dada pela

- chave privada de **A**, **KA**- inferindo assim, que a mesma foi cifrada utilizando chave pública de **A**, **PUA**;

- e pela chave pública de **B**, **PUB** - inferindo que a chave privada de **B**, **KB** - também fez parte do seu processo de criação

partiu, de fato, do nodo que gerou o valor aleatório que resulta na chave secreta descriptografada por **A**. O ProVerif prova que o protocolo é capaz sim de garantir a correspondência entre os eventos e o conjunto de informações por eles manipulado em nível não injetável. Entretanto, através da consulta de nível injetável (rótulo "NAJMUS-4"), a ferramenta trouxe como inválida tal correspondência.

Código 35 – Najmus Saqib: Consultas de Autenticidade - 5 e 6

```
%(*"NAJMUS-5"*)
query GX': secretkey, KAXPUB': g, X': number, KB': skey;
    event(decryptedUsingKB(GX',KB')) ==>
        event(sendKAXPUB(KAXPUB')) && GX' =
            calcSecretKey(X',GENERATORPOINT)
                && KAXPUB' =
                    encryption(addsSkeyPlusNumber(KA, X'),
                        calcPublicKey(KB', GENERATORPOINT))
                            && KB' = KB.
%(*"NAJMUS-6"*)
query GX': secretkey, KAXPUB': g, X': number, KB': skey;
    event(decryptedUsingKB(GX',KB')) ==>
        inj-event(sendKAXPUB(KAXPUB')) && GX' =
            calcSecretKey(X',GENERATORPOINT) && KAXPUB'
                        = encryption(addsSkeyPlusNumber(KA, X'),
                            calcPublicKey(KB', GENERATORPOINT))
                                && KB' = KB.
```

A quinta verificação (rótulo "NAJMUS-5"), por sua vez, d (que teve seu sigilo comprovado anteriormente neste documento). Como resultado da análise, foi possível provar a garantia de autenticidade em nível não injetável. A sexta verificação (rótulo "NAJMUS-6") objetivava provar a garantia das correspondências acima listadas em nível injetável, porém, o ProVerif não consegui chegar a uma conclusão sobre a consulta em questão.

Código 36 – Najmus Saqib: Consultas de Autenticidade - 7 e 8

```
%(*"NAJMUS-7"*)
query GX': secretkey, KAXPUB': g, X': number, KB': skey;
    event(decryptedUsingKB(GX',KB')) ==>
        event(sendKAXPUB(KAXPUB')) && GX' =
            calcSecretKey(X',GENERATORPOINT) && KAXPUB'
                        = encryption(addsSkeyPlusNumber(KA, X'),
                            calcPublicKey(KB', GENERATORPOINT))
                                && KB' = KB.

%(*"NAJMUS-8"*)
query YX': secretkey, KBYPUA': g, Y': number, KA': skey;
    event(decryptedUsingKA(YX',KA')) ==>
```

```
13          inj-event(sendKBYPUA(KBYPUA')) && YX' =
14              calcSecretKey(Y',GENERATORPOINT) && KBYPUA'
15                          = encryption(addsSkeyPlusNumber(KB, Y'),
16                              calcPublicKey(KA', GENERATORPOINT))
17                                  && KA' = KA.
```

A sétima verificação (rótulo "NAJMUS-7") objetivava provar que o valor da chave simétrica calculada por **A** contém, de fato, a chave secreta calculada pelo nodo **B** e que esta chave secreta foi gerada a partir do valor aleatório **Y** (que teve seu sigilo comprovado anteriormente neste documento). Como resultado da análise, foi possível provar a garantia de autenticidade em nível não injetável. A oitava verificação (rótulo "NAJMUS-8") visava provar a garantia das correspondências acima listadas em nível injetável, porém, o ProVerif não consegui chegar a uma conclusão sobre a consulta em questão.

6.2.3.1 Sinopse

Assim, representando de forma mais compacta, fica provado que o protocolo proposto em (SAQIB, 2016) é capaz de garantir a manutenção do sigilo de todos os dados essenciais para seu correto funcionamento, assim como mostra a tabela 5.

Tabela 5 – Najmus Saqib: Sigilo

	KA	**KB**	**GENERATORPOINT**	**X**	**Y**
Sigilo	ok	ok	ok	ok	ok

Já com relação à autenticidade entre as mensagens trocadas entre **A** e **B**, ficou provada a correspondência, em nível de acordo não injetável, que descreve a descriptografia da chave recebida por **B**, e que o processo que a criou envolveu as chaves públicas de **B** e de **A**.

Ficou ficou provada também a correspondência, em nível de acordo não injetável, que descreve a descriptografia da chave recebida por **A**, e que o processo que a criou envolveu as chaves públicas de **A** e de **B**.

Provou-se também que, em nível de acordo não injetável, que a correspondência dos eventos que levam ao cálculo da chave simétrica por **B** contém, de fato, uma chave secreta calculada por **A** e que foi gerada a partir de **X**.

Por fim, ficou provado, em nível de acordo não injetável, que a correspondência dos eventos que levam ao cálculo da chave simétrica

por **A** contém, de fato, uma chave secreta calculada por **B** e que foi gerada a partir de **Y**. A tabela 6 organiza os resultado das consultas previamente apresentadas, onde **ok** representa a garantia da consulta, **x** a não validade da consulta e **?** a não capacidade da ferramenta em definir o resultado.

Tabela 6 – Najmus Saqib: Autenticação

NAJMUS-	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Não Injetável	ok		ok		ok		ok	
Injetável		x		x		?		?

7 CONSIDERAÇÕES FINAIS

O aumento da popularidade de redes de sensores sem fio está fortemente atrelado ao destaque na última década que sistemas e dispositivos *IoT* ganharam da indústria e do público em geral. Entretanto, a preocupação dos que desenvolvem estes tipos de sistemas e dispositivos com a questão da segurança das comunicações de dados não parece ter a mesma força. Como resultado, frequentemente novos ataques de segurança são noticiados mundialmente ao envolverem o abuso de brechas de segurança.

Ferramentas, como protocolos de segurança, visam fornecer segurança as comunicações que ocorrem entre participantes de redes de computadores. Por não serem de fácil projeto, o processo de desenvolvimento de protocolos é propenso a erros, podendo estes virem a ser exploradas por agentes mal intencionados.

Com o objetivo de eliminar estas falhas de projeto e fornecer aos desenvolvedores garantias do que os protocolos criptográficos são ou não capazes de garantir quanto às propriedades de segurança, verificações de corretude utilizando métodos formais podem ser utilizadas.

Neste trabalho foram escolhidos três protocolos de segurança voltados a redes de sensores sem fio para serem formalmente verificados quanto às propriedades de sigilo e autenticidade. Todos foram capazes de garantir a manutenção do sigilo das informações chave para seus corretos funcionamentos, entretanto, com relação à autenticidade entre participantes, os resultados diferiram de protocolo para protocolo.

Baseado no trabalho de Oliveira et al. (2008), se modelou o acordo de chaves entre dois participantes (chamados de **A** e **B**) de mesma hierarquia, oito consultas foram formuladas para avaliar a correspondência de eventos dentro do protocolo.

A primeira delas (*"TPBC-1"*) verificou se a chave utilizada na descriptografia da mensagem recebida por **B** era a mesma que, teoricamente, foi utilizada para cifrar a mensagem em um evento ocorrido anteriormente dentro do protocolo por **B**. Como resultado, provou-se que esta correspondência é verdadeira em nível não injetável e também em nível injetável, resultado da segunda consulta (*"TPBC-2"*).

A terceira (*"TPBC-3"*) e quarta *"TPBC-4"* consultas tem papel similar ao das duas primeiras consultas, porém, desta vez verificando a relação da chave utilizada por **A** é a mesma que, teoricamente, foi utilizada para cifrar a mensagem em um evento ocorrido anteriormente dentro do protocolo por **B**. Como resultado, provou-se que esta cor-

respondência é verdadeira em nível não injetável e também em nível injetável, resultado da consulta *"TPBC-4"*.

A consulta *"TPBC-5"* tratava de verificar se a chave simétrica utilizada por **B**, que se acreditava ter partido de **A**, realmente utilizou a chave privada de **A** em sua criação. Obteve-se a prova de que este acordo é valido tanto em nível não injetável, como também em nível injetável *"TPBC-6"*.

A consulta *"TPBC-7"*, por fim, tratava de verificar se a chave simétrica utilizada por **A**, que se acreditava ter partido de **B**, realmente utilizou a chave privada de **B** em sua criação. Obteve-se a prova de que este acordo é valido tanto em nível não injetável, como também em nível injetável *"TPBC-8"*.

Já no protocolo baseado no texto de Herrera e Hu (2012), entretanto, o resultado das correspondências não foram tão homogêneas como as vistas na análise do trabalho de Oliveira et al. (2008). A verificação *"HERRERA-1"* tratava de verificar se a chave descriptografada por **S1** de fato partiu de um evento que ocorreu no protocolo. O resultado da consulta provou que a correspondência é válida em nível não injetável, mas trouxe como inválida a correspondência em nível de acordo injetável, resultado da consulta *"HERRERA-2"*.

A consulta *"HERRERA-3"*, por sua vez, tratava de verificar a criação da chave por parte da estação base de fato é resultado do envio do **id** e do ***nonce*** do nodo **S1** do qual se pensa estar recebendo estes dados. Como resultado, o ProVerif provou que a correspondência não é válida em nenhum dos níveis de injetividade, visto que ambos os valores tem seu sigilo violado durante a execução do protocolo.

Já a consulta *"HERRERA-4"* tinha como objetivo verificar se a chave descriptografada por **S1** utilizando sua chave privada é resultado de um evento passado previsto no protocolo que cifrava a chave simétrica utilizando a chave pública de **S1**. Ficou provada a correspondência em nível de acordo não injetável, mas a ferramenta não foi capaz de apresentar um resultado quanto ao nível injetável (*"HERRERA-5"*).

Em *"HERRERA-6"*, por fim, foi verificado se o valor *hash* obtido pelo nodo **S1** baseado na chave descriptografada foi, de fato, assinado pela estação base e se esta chave foi criada a partir dos dados que o nodo **S1** havia enviado anteriormente à estação base. Ficou provado que o protocolo é capaz de garantir a manutenção de um acordo em nível não injetável, porém a ferramenta não foi capaz de chegar à uma conclusão sobre acordos injetáveis (*"HERRERA-7"*).

No último dos protocolos, baseado no texto de Saqib (2016), começou-se verificando se a chave recebida por **B** é resultado de um

processo de criptografia que envolvia a chave pública de **B** e a chave privada de **A** (*"NAJMUS-1"*). A ferramenta provou que o protocolo é capaz de garantir a manutenção da correspondência de acordos em nível não injetável, mas trouxe que a correspondência de acordos em nível injetável não são válidos (*"NAJMUS-2"*).

Em *"NAJMUS-3"* verificou se a chave recebida por **A** é resultado de um processo de criptografia que envolvia a chave pública de **A** e a chave privada de **B**. A ferramenta provou que o protocolo é capaz de garantir a manutenção da correspondência de acordos em nível não injetável, mas trouxe que a correspondência de acordos em nível injetável não são válidos (*"NAJMUS-4"*).

Já em *"NAJMUS-5"*, o objetivo era provar que a chave simétrica calculada por **B** é resultado da utilização da chave secreta de **A** e que esta chave continha o valor aleatório **X** escolhido por **A**. O ProVerif foi capaz de provar a garantia da correspondência dos eventos em nível não injetável, mas não conseguiu chegar a uma conclusão quanto aos acordos de nível injetável (*"NAJMUS-6"*).

Já em *"NAJMUS-7"*, o objetivo era provar que a chave simétrica calculada por **A** é resultado da utilização da chave secreta de **B** e que esta chave continha o valor aleatório **Y** escolhido por **B**. O ProVerif foi capaz de provar a garantia da correspondência dos eventos em nível não injetável, mas não conseguiu chegar a uma conclusão quanto aos acordos de nível injetável (*"NAJMUS-8"*).

Com as provas de sigilo e autenticidade alcançadas sobre os três protocolos neste trabalho analisados, desenvolvedores podem avaliar quais deles melhor se adaptam às suas necessidades, visto que, como discutido anteriormente, segurança não é algo único, variando de aplicação para aplicação.

Como trabalhos futuros se propõe a realização da verificação formal dos protocolos acima fazendo uso, desta vez, de ferramentas que adotem o modelo computacional ao invés do modelo simbólico. Desta forma, reduz-se o nível de abstração empregado e, consequentemente, aproxima ainda mais da realidade o funcionamento dos protocolos neste trabalho analisados.

REFERÊNCIAS

ABADI, H.; NEEDHAM, R. Prudent engineering practice for cryptographic protocols. In: *Proceedings of 1994 IEEE Computer Society Symposium on Research in Security and Privacy*. IEEE Comput. Soc. Press, 1994. <https://doi.org/10.1109/risp.1994.296587>.

ABADI, M. Security protocols: Principles and calculi. In: *Foundations of Security Analysis and Design IV*. Springer Berlin Heidelberg, 2006. p. 1–23. $<https://doi.org/10.1007/978-3-540-74810-6_1>$.

AKYILDIZ, I. et al. Wireless sensor networks: a survey. *Computer Networks*, Elsevier BV, v. 38, n. 4, p. 393–422, mar 2002. <https://doi.org/10.1016/s1389-1286(01)00302-4>.

ALCARAZ, C.; ROMAN, R.; LOPEZ, J. Análisis de primitivas criptográficas para redes de sensores. In: *VI Jornadas de Ingeniería Telemática (JITEL'07)*. Málaga (Spain): [s.n.], 2007. p. 401–408. ISBN 978-84-690-6670-6. <http://www.telematica.ws/jitel/2007/>.

ANGLUIN, D.; FRAZIER, M.; PITT, L. Learning conjunctions of horn clauses. *Machine Learning*, v. 9, n. 2, p. 147–164, Jul 1992. ISSN 1573-0565.

ARAPINIS, M.; CHEVAL, V.; DELAUNE, S. Composing security protocols: from confidentiality to privacy. In: FOCARDI, R.; MYERS, A. (Ed.). *Proceedings of the 4th International Conference on Principles of Security and Trust (POST'15)*. London, UK: Springer, 2015. (Lecture Notes in Computer Science, v. 9036), p. 324–343. <http://www.lsv.ens-cachan.fr/Publis/PAPERS/PDF/ACD-post15.pdf>.

AVALLE, M.; PIRONTI, A.; SISTO, R. Formal verification of security protocol implementations: a survey. *Formal Aspects of Computing*, Springer Nature, v. 26, n. 1, p. 99–123, dec 2012. <https://doi.org/10.1007/s00165-012-0269-9>.

BECHER, A.; BENENSON, Z.; DORNSEIF, M. Tampering with motes: Real-world physical attacks on wireless sensor networks. In: _____. *Security in Pervasive Computing: Third International Conference, SPC 2006, York, UK, April 18-21, 2006. Proceedings*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006. p. 104–118.

BELLARE, M.; ROGAWAY, P. Random oracles are practical: A paradigm for designing efficient protocols. In: *Proceedings of the 1st ACM Conference on Computer and Communications Security*. New York, NY, USA: ACM, 1993. (CCS '93), p. 62–73. ISBN 0-89791-629-8. <http://doi.acm.org/10.1145/168588.168596>.

BLANCHET, B. Modeling and verifying security protocols with the applied pi calculus and proverif. *Foundations and Trends® in Privacy and Security*, v. 1, n. 1-2, p. 1–135, 2016. ISSN 2474-1558. <http://dx.doi.org/10.1561/3300000004>.

BLANCHET, B. *ProVerif: Cryptographic protocol verifier in the formal model*. 2017. <http://prosecco.gforge.inria.fr/personal/bblanche/proverif/>.

BLANCHET, B. et al. *ProVerif 1.97pl1: Automatic Cryptographic Protocol Verifier, User Manual and Tutorial*. [S.l.], sep 2017.

BONEH, D.; SHPARLINSKI, I. E. On the unpredictability of bits of the elliptic curve diffie-hellman scheme. In: _____. *Advances in Cryptology — CRYPTO 2001: 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19–23, 2001 Proceedings*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001. p. 201–212. ISBN 978-3-540-44647-7.

BURSUC, S. Secure two-party computation in applied pi-calculus: Models and verification. In: *Trustworthy Global Computing*. [S.l.]: Springer International Publishing, 2016. p. 1–15.

CARLE, G. *Network Security Chapter 7 Cryptographic Protocols*. mar 2003.

CERRUDO, C. *An Emerging US (and World) Threat: Cities Wide Open to Cyber Attacks*. 2015. $<https://ioactive.com/pdfs/IOActive_HackingCitiesPaper_cyber-security_CesarCerrudo.pdf>$.

CHANG, R.; SHMATIKOV, V. *Formal Analysis of Authentication in Bluetooth Device Pairing*. 2007.

CHASE, J. *The Evolution of the Interne of Things*. 2013. <http://www.ti.com/lit/ml/swrb028/swrb028.pdf>.

CHONG, H. F.; NG, D. W. K. Development of iot device for traffic management system. In: *2016 IEEE Student Conference on Research and Development (SCOReD)*. [S.l.: s.n.], 2016. p. 1–6.

CORBELLINI, A. *Elliptic Curve Cryptography: a gentle introduction.* may 2015. $<http://andrea.corbellini.name/2015/05/17/elliptic-curve-cryptography-a-gentle-introduction/>$.

DELAUNE, S.; KREMER, S.; RYAN, M. D. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, IOS Press, v. 17, n. 4, p. 435–487, jul. 2009.

DELAUNE, S. et al. A formal analysis of authentication in the TPM. In: DEGANO, P.; ETALLE, S.; GUTTMAN, J. (Ed.). *Revised Selected Papers of the 7th International Workshop on Formal Aspects in Security and Trust (FAST'10)*. Pisa, Italy: Springer, 2010. (Lecture Notes in Computer Science, v. 6561), p. 111–125.

DESAI, D. *IoT devices in the enterprise.* nov 2016. <http://www.csoonline.com/article/3142484/internet-of-things/report-surveillance-cameras-most-dangerous-iot-devices-in-enterprise.html>.

DIAZ, J.; ARROYO, D.; RODRIGUEZ, F. B. On securing online registration protocols: Formal verification of a new proposal. *Knowledge-Based Systems*, v. 59, n. Supplement C, p. 149 – 158, 2014. ISSN 0950-7051.

DIFFIE, W.; HELLMAN, M. New directions in cryptography. *IEEE Transactions on Information Theory*, Institute of Electrical and Electronics Engineers (IEEE), v. 22, n. 6, p. 644–654, nov 1976. <https://doi.org/10.1109/tit.1976.1055638>.

DOLEV, D.; YAO, A. On the security of public key protocols. *IEEE Trans. Inf. Theor.*, IEEE Press, Piscataway, NJ, USA, v. 29, n. 2, p. 198–208, set. 1983. ISSN 0018-9448. <http://dx.doi.org/10.1109/TIT.1983.1056650>.

EVANS, D. L.; BOND, P. J.; BEMENT, J. A. L. *Standards for security categorization of federal information and information systems.* [S.l.], feb 2004. <https://doi.org/10.6028/nist.fips.199>.

GAZIS, V. et al. Short paper: IoT: Challenges, projects, architectures. In: *2015 18th International Conference on Intelligence in Next Generation Networks.* IEEE, 2015. <https://doi.org/10.1109/icin.2015.7073822>.

GRAH, J. S. *Hash Functions in Cryptography.* Dissertação (Mestrado) — Universitet i Bergen, jun 2008.

GUPTA, R. et al. Security for wireless sensor networks in military operations. In: *2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT)*. [S.l.: s.n.], 2013. p. 1–6.

GUTTMAN, B.; ROBACK, E. A. *SP 800-12. An Introduction to Computer Security: The NIST Handbook*. Gaithersburg, MD, United States, 1995.

HERRERA, A.; HU, W. A key distribution protocol for wireless sensor networks. In: *37th Annual IEEE Conference on Local Computer Networks*. [S.l.: s.n.], 2012. p. 140–143. ISSN 0742-1303.

HIRSCHI, L.; BAELDE, D.; DELAUNE, S. A method for verifying privacy-type properties: the unbounded case. In: LOCASTO, M.; SHMATIKOV, V.; ERLINGSSON, Ú. (Ed.). *Proceedings of the 37th IEEE Symposium on Security and Privacy (S&P'16)*. San Jose, California, USA: IEEECSP, 2016. p. 564–581.

ISLAM, S. Security property validation of the sensor network encryption protocol (SNEP). *Computers*, MDPI AG, v. 4, n. 3, p. 215–233, jul 2015. <https://doi.org/10.3390/computers4030215>.

ITU. *X.800 : Security architecture for Open Systems Interconnection for CCITT Applications*. mar 1991.

KADRI, B. et al. An efficient key management scheme for hierarchical wireless sensor networks. *Wireless Sensor Network*, Scientific Research Publishing, Inc,, v. 04, n. 06, p. 155–161, 2012. <https://doi.org/10.4236/wsn.2012.46022>.

KASRAOUI, M.; CABANI, A.; CHAFOUK, H. Formal verification of wireless sensor key exchange protocol using avispa. In: *2014 International Symposium on Computer, Consumer and Control*. [S.l.: s.n.], 2014. p. 387–390.

KHAN, S. F. Health care monitoring system in internet of things (iot) by using rfid. In: *2017 6th International Conference on Industrial Technology and Management (ICITM)*. [S.l.: s.n.], 2017. p. 198–204.

KINGSLY, S.; CHANDRAN, G. C. Critical study on constraints in wireless sensor network applications. *International Journal of Engineering Research & Technology (IJERT)*, v. 2, n. 7, jul 2013.

KLEBERGER, P.; MOULIN, G. Short paper: Formal verification of an authorization protocol for remote vehicle diagnostics. In: *2013 IEEE Vehicular Networking Conference*. [S.l.: s.n.], 2013. p. 202–205. ISSN 2157-9857.

KOROLOV, M. *Report: Surveillance cameras most dangerous IoT devices in enterprise*. nov 2016. <http://www.csoonline.com/article/3142484/internet-of-things/report-surveillance-cameras-most-dangerous-iot-devices-in-enterprise.html>.

KRANENBURG, R. van; BASSI, A. IoT challenges. *Communications in Mobile Computing*, Springer Nature, v. 1, n. 1, p. 9, 2012. <https://doi.org/10.1186/2192-1121-1-9>.

KREMER, S. *Formal Verification of Cryptographic Protocols*. jun. 2006. Invited tutorial, 7th School on Modelling and Verifying Parallel Processes (MOVEP'06), Bordeaux, France. 5 pages.

LEE, I.; LEE, K. The internet of things (IoT): Applications, investments, and challenges for enterprises. *Business Horizons*, Elsevier BV, v. 58, n. 4, p. 431–440, jul 2015. <https://doi.org/10.1016/j.bushor.2015.03.008>.

LOPEZ, J.; ZHOU, J. *Wireless Sensor Network Security*. Amsterdam, The Netherlands, The Netherlands: IOS Press, 2008. ISBN 1586038133, 9781586038137.

LOUREIRO, A. A. et al. Redes de sensores sem fio. In: *Simpósio Brasileiro de Redes de Computadores*. [S.l.: s.n.], 2003. v. 21, p. 19–23.

LOUREIRO, A. A. a. Redes de sensores sem fio. *RBCM - J. of the Brazilian Soc. Mechanical Sciences*, v. 19, n. 3, p. 332–340, 1997.

LOWE, G. An attack on the needham-schroeder public-key authentication protocol. *Inf. Process. Lett.*, Elsevier North-Holland, Inc., Amsterdam, The Netherlands, The Netherlands, v. 56, n. 3, p. 131–133, nov. 1995. ISSN 0020-0190. <http://dx.doi.org/10.1016/0020-0190(95)00144-2>.

LOWE, G. A hierarchy of authentication specifications. In: *Proceedings of the 10th IEEE Workshop on Computer Security Foundations*. Washington, DC, USA: IEEE Computer Society, 1997. (CSFW '97), p. 31–. ISBN 0-8186-7990-5. <http://dl.acm.org/citation.cfm?id=794197.795075>.

MARNE, M.; PATIL, S. The constraints in wireless sensor network - a review. *IJSRD (International Journal of Scientific Research and Development)*, jan 2016.

MARTINS, D.; GUYENNET, H. Wireless sensor network attacks and security mechanisms: A short survey. In: *2010 13th International Conference on Network-Based Information Systems*. [S.l.: s.n.], 2010. p. 313–320. ISSN 2157-0418.

MILNER, R.; PARROW, J.; WALKER, D. A calculus of mobile processes, i. *Inf. Comput.*, Academic Press, Inc., Duluth, MN, USA, v. 100, n. 1, p. 1–40, set. 1992. ISSN 0890-5401. <http://dx.doi.org/10.1016/0890-5401(92)90008-4>.

NAWIR, M. et al. Internet of things (IoT): Taxonomy of security attacks. In: *2016 3rd International Conference on Electronic Design (ICED)*. IEEE, 2016. <https://doi.org/10.1109/iced.2016.7804660>.

NEEDHAM, R. M.; SCHROEDER, M. D. Using encryption for authentication in large networks of computers. *Commun. ACM*, ACM, New York, NY, USA, v. 21, n. 12, p. 993–999, dez. 1978. ISSN 0001-0782. <http://doi.acm.org/10.1145/359657.359659>.

NGUYEN, K. T.; OUALHA, N.; LAURENT, M. Authenticated key agreement mediated by a proxy re-encryptor for the internet of things. In: _____. *Computer Security – ESORICS 2016: 21st European Symposium on Research in Computer Security, Heraklion, Greece, September 26-30, 2016, Proceedings, Part II*. Cham: Springer International Publishing, 2016. p. 339–358. ISBN 978-3-319-45741-3.

NOGUEIRA, R. B. *Verificação Formal de Protocolos Criptográficos - O Caso dos Protocolos em Cascata*. Dissertação (Mestrado) — Universidade de Brasília, 2008.

OLIVEIRA, L. B. et al. Tinypbc: Pairings for authenticated identity-based non-interactive key distribution in sensor networks. *Comput. Commun.*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 34, n. 3, p. 485–493, mar. 2011. ISSN 0140-3664. <http://dx.doi.org/10.1016/j.comcom.2010.05.013>.

OLIVEIRA, L. B. et al. Tinypbc: Pairings for authenticated identity-based non-interactive key distribution in sensor networks. p. 173–180, June 2008.

OTT, D. E. IoT security through the lens of energy efficiency: Energy as a first-order security consideration. In: *2016 Cybersecurity Symposium (CYBERSEC)*. IEEE, 2016. <https://doi.org/10.1109/cybersec.2016.011>.

O'SHEA, N. *Verification and Validation of Security Protocol Implementations*. Dissertação (Mestrado) — University of Edinburgh, 2010.

PADMAVATHI, G.; SHANMUGAPRIYA, D. A survey of attacks, security mechanisms and challenges in wireless sensor networks. *CoRR*, abs/0909.0576, 2009. <http://arxiv.org/abs/0909.0576>.

PIVA, F. R. *Verificação formal de protocolos de trocas justas utilizando o método de espaços de fitas*. Dissertação (Mestrado) — Universidade Estadual de Campinas, 2009.

RANI, A.; KUMAR, S. A survey of security in wireless sensor networks. In: *2017 3rd International Conference on Computational Intelligence & Communication Technology (CICT)*. IEEE, 2017. <https://doi.org/10.1109/ciact.2017.7977334>.

RIVEST, R. L.; SHAMIR, A.; ADLEMAN, L. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, ACM, New York, NY, USA, v. 21, n. 2, p. 120–126, fev. 1978. ISSN 0001-0782. <http://doi.acm.org/10.1145/359340.359342>.

RONEN, E. et al. Iot goes nuclear: Creating a zigbee chain reaction. In: . [S.l.: s.n.], 2016.

ROSE, K.; ELDRIDGE, S.; CHAPIN, L. *The Internet of Things (IoT): An Overview*. 2015. <https://www.internetsociety.org/sites/default/files/ISOC-IoT-Overview-20151221-en.pdf>.

RYAN, M. D.; SMYTH, B. Applied pi calculus. In: CORTIER, V.; KREMER, S. (Ed.). *Formal Models and Techniques for Analyzing Security Protocols*. IOS Press, 2011. cap. 6. <http://www.bensmyth.com/files/Smyth10-applied-pi-calculus.pdf>.

SANTOS, E. dos. *Formalização e verificação de um protocolo de autenticação multifator*. Dissertação (Mestrado) — Universidade Federal de Santa Catarina, 2012.

SAQIB, N. Key exchange protocol for wsn resilient against man in the middle attack. In: *2016 IEEE International Conference on Advances in Computer Applications (ICACA)*. [S.l.: s.n.], 2016. p. 265–269.

SHAIKH, R.; AND, A. R.; DEVANE, S. Formal verification of payment protocol using avispa. v. 3, 09 2010.

SHEN, G. et al. An extended UML method for the verification of security protocols. In: *2014 19th International Conference on Engineering of Complex Computer Systems*. IEEE, 2014. <https://doi.org/10.1109/iceccs.2014.12>.

SHIREY, D. R. *Internet Security Glossary*. RFC Editor, maio 2000. RFC 2828. (Request for Comments, 2828). <https://rfc-editor.org/rfc/rfc2828.txt>.

SIMMONS, G. J. Symmetric and asymmetric encryption. *ACM Comput. Surv.*, ACM, New York, NY, USA, v. 11, n. 4, p. 305–330, dez. 1979. ISSN 0360-0300. <http://doi.acm.org/10.1145/356789.356793>.

STALLINGS, W. *Cryptography and Network Security: Principles and Practice*. 5th. ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2010. ISBN 0136097049, 9780136097044.

SULLIVAN, N. *A (relatively easy to understand) primer on elliptic curve cryptography*. oct 2013. <https://arstechnica.com/information-technology/2013/10/a-relatively-easy-to-understand-primer-on-elliptic-curve-cryptography/>.

TRIPATHI, R.; AGRAWAL, S. Comparative study of symmetric and asymmetric cryptography techniques. *International Journal of Advance Foundation and Research in Computer (IJAFRC)*, v. 1, jun 2014.

V.HEMAMALINI et al. A survey on elementary, symmetric and asymmetric key cryptographic techniques. *International Journal of Computing Academic Research (IJCAR)*, v. 5, feb 2016.

VIGO, R.; FILè, G. Expressive power of definite clauses for verifying authenticity. *2009 22nd IEEE Computer Security Foundations Symposium (CSF)*, IEEE Computer Society, Los Alamitos, CA, USA, v. 00, p. 251–265, 2009. ISSN 1063-6900.

WANG, Y.; ATTEBURY, G.; RAMAMURTHY, B. A survey of security issues in wireless sensor networks. *CSE*, 2006.

WOO, E. *The RSA Encryption Algorithm (2 of 2: Generating the Keys)*. oct 2014. <https://www.youtube.com/watch?v=oOcTVTpUsPQ>.

WOO, T. Y. C.; LAM, S. S. A semantic model for authentication protocols. In: *Proceedings 1993 IEEE Computer Society Symposium on Research in Security and Privacy*. [S.l.: s.n.], 1993. p. 178–194.

XIAO, Y. *Security in Distributed, Grid, Mobile, and Pervasive Computing*. Boston, MA, USA: Auerbach Publications, 2007. ISBN 0849379210.

XU, W. et al. Jamming sensor networks: attack and defense strategies. *IEEE Network*, v. 20, n. 3, p. 41–47, May 2006. ISSN 0890-8044.

YANG, S. et al. A new design of security wireless sensor network using efficient key management scheme. In: *2010 2nd IEEE InternationalConference on Network Infrastructure and Digital Content*. IEEE, 2010. <https://doi.org/10.1109/icnidc.2010.5657820>.

YICK, J.; MUKHERJEE, B.; GHOSAL, D. Wireless sensor network survey. *Computer Networks*, v. 52, n. 12, p. 2292 – 2330, 2008. ISSN 1389-1286. <http://www.sciencedirect.com/science/article/pii/S1389128608001254>.

ZIA, T.; ZOMAYA, A. Security issues in wireless sensor networks. In: *Systems and Networks Communications, 2006. ICSNC '06. International Conference on*. [S.l.: s.n.], 2006. p. 40–40.