Universidade Federal do Amazonas
Instituto de Computação
Programa de Pós-Graduacao em Informática

Gabriel Carrijo Bento Teixeira

Um Data Diode com Hardware Criptográfico para Redes Industriais Críticas

Manaus
Dezembro de 2017

Gabriel Carrijo Bento Teixeira

Um Data Diode com Hardware Criptográfico para Redes Industriais Críticas

Trabalho apresentado ao Programa de Pós-Graduação em Informática do Instituto de Computação da Universidade Federal do Amazonas como requisito final para obtenção do grau de Mestre em Informática.

Orientador: Prof. Dr. Eduardo Luzeiro Feitosa

Manaus
Dezembro de 2017

Ficha Catalográfica

Ficha catalográfica elaborada automaticamente de acordo com os dados fornecidos pelo(a) autor(a).

Teixeira, Gabriel Carrijo Bento
T266u Um Data Diode com Hardware Criptográfico para Redes Industriais Críticas / Gabriel Carrijo Bento Teixeira. 2017
66 f.: 31 cm.

Orientador: Eduardo Luzeiro Feitosa
Dissertação (Mestrado em Informática) - Universidade Federal do Amazonas.

1. Data Diode. 2. Gateway Unidirecional. 3. Rede Crítica. 4. Hardware Criptográfico. 5. Comunicação Segura. I. Feitosa, Eduardo Luzeiro II. Universidade Federal do Amazonas III. Título

FOLHA DE APROVAÇÃO

"Um Data Diode com Hardware Criptográfico para Redes Industriais Críticas"

GABRIEL CARRIJO BENTO TEIXEIRA

Dissertação de Mestrado defendida e aprovada pela banca examinadora constituída pelos Professores:

Eduardo Luzeiro Feitosa

Prof. Eduardo Luzeiro Feitosa - PRESIDENTE

Horácio A. B. Fernandes de Oliveira

Prof. Horácio Antonio B. Fernandes de Oliveira - MEMBRO INTERNO

Thiago Gomes Rodrigues

Prof. Thiago Gomes Rodrigues - MEMBRO EXTERNO

Manaus, 15 de Dezembro de 2017

Resumo

Redes Industriais são ambientes altamente sensíveis do ponto de vista da Segurança da Informação, visto que incidentes computacionais podem ocasionar prejuízos incalculáveis. Com o passar dos anos a interligação dessas redes com ambientes corporativos e, consequentemente a Internet, trouxe sérias preocupações sobre a integridade das informações e equipamentos envolvidos. Diversas soluções têm sido propostas com o objetivo de proteger infraestruturas de comunicação de dados em Redes Industriais. Contudo, será que realmente é possível ou factível garantir que as soluções implementadas são realmente seguras? Neste sentido, esta dissertação apresenta um esquema de segurança capaz de tratar os problemas encontrados na integração de redes industriais críticas com redes corporativas inseguras, objetivando garantir integridade dos dados e confiabilidade entre os dispositivos. Para esse fim é proposta a utilização de um *Data Diode* na interligação das redes para a proteção da planta industrial e um hardware criptográfico TPM (*Trusted Platform Module*) para garantia de integridade e confiabilidade dos dispositivos envolvidos. Como forma de provar a efetividade dessa arquitetura, foram realizados testes, que ao final no trabalho, mostram que é possível alcançar resultados superiores aos trabalhos já existentes na literatura.

Palavras-chave: Redes Industriais, Data Diode, Segurança da Informação, TPM.

Abstract

Industrial networks are highly sensitive environments from the point of view of information security with a view to computer incidents can cause incalculable damage. Over the years the connection of those networks with enterprise environments and consequently the Internet, has brought serious concerns about the integrity of the information and equipment involved. Several solutions have been proposed with the aim of protecting Industrial Networks in data communications infrastructure. However, is it really possible or feasible to ensure that the solutions implemented are really safe? In this sense, this work presents a security scheme able to deal with the problems encountered in the integration of critical industrial networks with insecure corporate networks, aiming to ensure data integrity and reliability being the devices. To this end it is proposed to use a *Data Diode* in the interconnection of networks for the protection of industrial plant and a cryptographic hardware TPM (*Trusted Platform Module*) to guarantee integrity and reliability of the devices involved. In order to prove the effectiveness of this architecture, tests were carried out to the end the work show that it is possible to achieve better results than those existing in the literature.

Keywords: Industrial Networks, Data Diode, Information Security, TPM.

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo 1

Introdução

Desde seu surgimento, os sistemas de automação e controle industriais em ambientes de infraestrutura crítica (geração de energia, telecomunicações, água e esgoto, transporte coletivo, óleo e gás, por exemplo) possuíam sistemas operacionais e protocolos de comunicações proprietários, nos quais apenas especialistas tinham conhecimento. Em geral, o domínio dessas soluções proprietárias estava restrito aos fabricantes de equipamentos. Assim, a segurança cibernética era realizada basicamente por isolamento e desconhecimento desses sistemas. Decorrente disso, a única preocupação com a segurança era controlar o acesso físico aos equipamentos de automação e controle [1].

Contudo, a evolução tecnológica permitiu que os sistemas de automação e controle industriais pudessem operar e se comunicar de forma eficiente entre os diferentes equipamentos de uma planta industrial. As empresas de infraestrutura crítica perceberam que era possível aperfeiçoar seu desempenho financeiro através da integração das operações via redes de computadores, o que impactou positivamente na melhoria da qualidade do serviço ofertado, na minimização do tempo de interrupção de serviços, na redução de custos de produção e no aumento da produtividade [2], tornando-as um grande caso de sucesso. Dessa forma, as empresas de infraestrutura crítica passaram a migrar seus equipamentos tradicionais (legados) para dispositivos mais inteligentes e interligados. Em outras palavras, os dados gerados nos sistemas de automação e controle industriais passaram a ser acessados por sistemas da rede corporativa através das tecnologias de rede e da Internet.

Essa nova integração trouxe uma série de benefícios, mas também potencializou os problemas de segurança nessas redes. Em linhas gerais, tais redes demandam alta disponibilidade na operação e, consequentemente, precisam ser protegidas contra incidentes deliberados e inadvertidos que podem comprometer seu funcionamento [3]. Assim, a integração desses dois ambientes (redes industrias e redes corporativas), do ponto de vista da Segurança da Informação, não é

uma tarefa simples.

1.1 Motivação

Embora os sistemas de automação e controle industriais tipicamente estejam separados das redes corporativas com acesso a Internet, a preocupação com esquemas de segurança específicos para esses cenários ganhou grande repercussão e fomento no meio acadêmico a partir de 2010, quando o Worm Stuxnet infectou várias industrias no Irã, inclusive uma planta de produção nuclear [2]. A partir desse ano, mais de 800 incidentes de segurança em redes industriais e mais de 600 vulnerabilidades já foram reportadas ao ICS-CERT (*Industrial Control Systems Cyber Emergency Response Team*) e desde 2012, em média, 200 novos incidentes são reportados por ano [4]. De acordo com a base de dados do Repositório de Incidentes de Segurança (RISI - *Repository of Industrial Security Incidents*) houveram 240 incidentes de segurança da informação relacionados a invasão a partir de dezembro de 2012 em todo o mundo [5]. Portanto, fica explícita a necessidade por buscar propostas eficazes de esquemas de Segurança da Informação voltados a essas redes.

No que tange soluções para o problema de segurança na interligação de redes em infraestruturas críticas, a primeira e mais óbvia é a segmentação de rede através de um dispositivo de roteamento ou dos tradicionais dispositivos de segurança (firewall, IDS/IPS, por exemplo). Contudo, ela de fato não representa uma solução eficiente e definitiva, especialmente contra ameaças desconhecidas (*zero day*) e falhas de configuração [3]. A segunda solução é o isolamento da rede crítica de outras redes, uma ideia simples e bastante utilizada. Porém, essa separação completa impossibilita a extração de informações em tempo real desses ambientes e a tomada rápida de decisão.

Mais recentemente, a interligação da rede crítica com outras redes através de um diodo de dados (*Data Diode*) passou a ser difundida em publicações científicas. Conhecida como *gateway* unidirecional, a ideia é permitir a transmissão dos dados em uma única direção, sendo impossível transferir dados de volta na direção oposta, a fim de proteger a informação e garantindo a confidencialidade dos dados.

Mas como garantir segurança e confidencialidade usando *Data Diode* se nenhuma sinalização, reconhecimento dos pacotes e outros mecanismos de segurança dos protocolos orientados à conexão são permitidos? Como confiar nas partes envolvidas na comunicação? Para solucionar esses problemas, estratégias como: (i) transferir os mesmos dados várias vezes, para diminuir a chance de perda e corrupção de dados [6]; (ii) aumentar o tamanho dos *buffers* de comunicação (kernel, rede), ajustando as prioridades do processo [7]; (iii) usar mecanismos de correção

de erro (FEC - *Forward Error Correction*) [8]; entre outras, foram propostas. Entretanto, elas falham tipicamente por não garantir a confiabilidade necessária à solução, nem a confidencialidade da informação.

Neste contexto, este trabalho apresenta um *Data Diode* capaz para aumentar a segurança em infraestruturas de redes críticas. O trabalho discute componentes e blocos de construção fundamentais (modelo funcional de implementação e implantação) no desenvolvimento de um *Data Diode* seguro.

1.2 Hipótese

A hipótese definida para este trabalho é a seguinte: Um *Data Diode* com um hardware criptográfico integrado para autenticação mútua e controle de retransmissão de dados em tempo real, utilizado como *gateway* de comunicação em infraestruturas críticas, possibilita a comunicação segura entre as redes envolvidas, garantindo a confiabilidade dos dispositivos e a integridade dos dados trafegados.

1.3 Objetivo

O objetivo deste trabalho é propor um esquema de segurança em infraestruturas de redes industriais críticas, através do uso de *Data Diodes* e hardware criptográfico, visando garantias de confiabilidade e integridade na transmissão de dados.

Especificamente, pretende-se:

- Avaliar como as funções de autenticação mútua geradas e gerenciadas por um hardware criptográfico podem ser empregadas em um cenário de comunicação unidirecional.
- Criar um protótipo funcional do esquema de segurança proposto para avaliar as garantias de confiabilidade e integridade na transmissão de dados. Serão testados vários cenários envolvendo diferentes arquiteturas de *Data Diodes*.

1.4 Contribuições Esperadas

A principal contribuição deste trabalho consiste em propor o uso de um *Data Diode* (*gateway* unidirecional) em conjunto com um hardware criptográfico (TPM - *Trusted Platform Module*) para restringir o fluxo e a comunicação no sentido rede insegura para rede crítica, a fim de garantir confiabilidade dos dispositivos envolvidos na comunicação.

1.5 Estrutura do Documento

Este documento está organizado em 5 Capítulos. No Capítulo 2 são apresentados os conceitos básicos necessários para a compreensão desta dissertação. O Capítulo 3 apresenta e discute alguns trabalhos relacionados que utilizam as características mencionadas no Capítulo anterior. O Capítulo 4 apresenta uma descrição detalhada do *Data Diode* proposto. Já o Capítulo 5 detalha a implementação e são apresentados os resultados que validam a investigação realizada nesta dissertação. Por fim, no Capítulo 6 são apresentadas as conclusões e os trabalhos futuros.

Capítulo 2

Conceitos Básicos

Este Capítulo apresenta os principais conceitos necessários para a compreensão dos temas abordados nesta dissertação, incluindo Sistemas de Controle Industriais, *Data Diode* e TPM.

2.1 Sistemas de Controle Industriais

Os Sistemas de Controle Industriais (ICS - *Industrial Control Systems*) são utilizados para automação de atividades relacionadas a infraestruturas críticas - geração de energia elétrica, gás, tratamento e fornecimento de água, combustíveis, transporte e outros setores da industria, por exemplo - a fim de prover interconexão entre os equipamentos industriais e os sistemas computacionais e, assim, obter informações e estabelecer controle de dispositivos como sensores, relés, ilhas de automação e todo tipo de equipamento com aplicações TCP/IP [4]. Sistemas de controle industriais também são conhecidos por Sistemas de Automação e Controle de Sistemas (IACS - *Industrial Automation Control Systems*) [5] ou Sistema de Controle de Supervisão e Aquisição de Dados (SCADA - *Supervisory Control And Data Acquisition*) [2].

Tradicionalmente, esses sistemas possuíam arquitetura completamente proprietária, o que dificultava a interoperabilidade entre dispositivos. Além disso, eram implantados em total isolamento de comunicação e segurança. Contudo, em 1997, o IEC (*International Electrotechnical Commission*) e o IEEE (*Institute of Electrical and Electronics Engineers*) começaram a trabalhar em conjunto na padronização dos ICS. Como resultado, a norma ISO/IEC 61850, um padrão que fornece uma arquitetura aberta a ser seguida, foi criada com o objetivo de garantir compatibilidade [2].

O problema é que a partir do momento em que foram padronizadas e, consequentemente, interligadas a redes inseguras (Figura 2.1), as ICS tornaram-se

alvos de ataques.

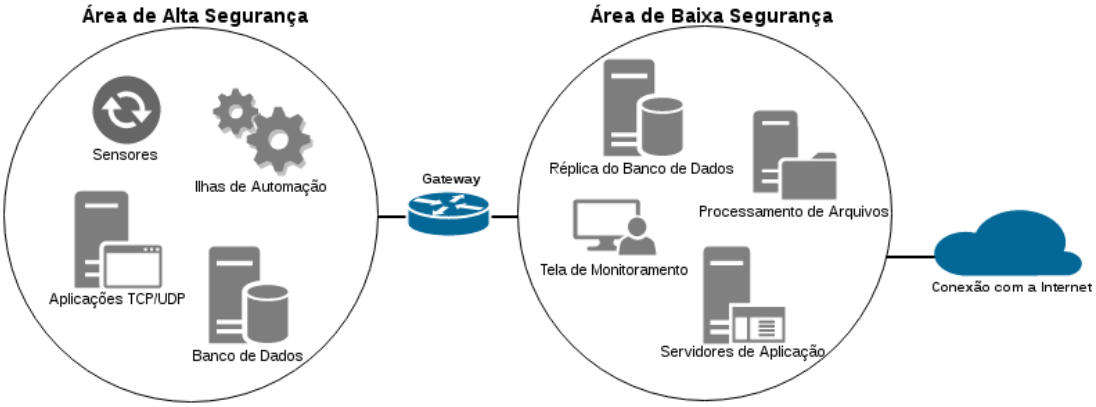

Figura 2.1: ICS integrado à rede corporativa

Elas não possuíam qualquer mecanismo de segurança que garantisse a integridade e confiabilidade dos dados, bem como sua proteção contra ameaças externas. Segundo Conklin [5], os dispositivos encontrados em um ICS não possuem qualquer implementação de segurança da informação lançada na última década, tornando esses ambientes parecidos com redes de 20 anos atrás, mas suscetíveis às ameaças de hoje.

O *trade-off* nesse cenário é que as indústrias tem a necessidade de integração dos ICS com aplicações internas, a fim de obter dados de produção para tomada de decisão em tempo real, mas com o mínimo de mudança na infraestrutura legada e com garantias de minimização dos riscos provenientes de uma conexão com a Internet e de redes corporativas. Em linhas gerais, as ICS precisam ter total confiabilidade, com o mínimo de paradas possíveis.

2.2 Soluções de Segurança para ICS

A partir dessa premissa surgiram diversos esquemas, implementações e modelos de segurança voltados para a proteção de infraestruturas críticas com o intuito de evitar novos ataques. Uma das técnicas utilizadas é o *air-gap*, onde os dispositivos mais críticos são completamente isolados, sem conexão com a Internet ou segmentos de rede conectados à Internet [5]. Embora amplamente utilizada, essa técnica pode ser violada quando a rede está aberta, por exemplo, para tarefas operacionais que necessitem de troca de dados. O worm Stuxnet, por exemplo, foi projetado para atacar justamente esse tipo de rede, se espalhando através de dispositivos de armazenamento móveis e infectando sistemas [5].

O uso de um *firewall* e um esquema de tolerância à intrusão baseado na detecção de intrusão, avaliação de impacto, determinação de estratégia de proteção e resposta em tempo real foi proposto por Hugang et al. [1]. Esse esquema consiste na detecção, avaliação e resposta a uma ameaça, sendo baseado em software e dependendo de uma base de assinaturas para detecção de ataques. Um estudo do CPNI (*Centre of the Protection of National Infrastructure*) concluiu que a melhor alternativa para a segregação das redes industriais, do ponto de vista da segurança, controle e escalabilidade são as redes baseadas em três zonas, tal qual uma DMZ (Zona Desmilitarizada) [3]. Essas zonas são segmentadas por dois firewall's, criando um terceiro ambiente entre o ICS e a rede corporativa onde há um sistema de centralização de dados que se comunica com as duas redes.

Fugindo das soluções tradicionais de segurança, autores como Okhravi e Sheldon [9], Oh et al. [10], Heo et al. [8] e Jeon e Na [4] propuseram a utilização de *Data Diodes* para a integração segura entre redes industriais e corporativas impedindo via hardware acessos indevidos em infraestruturas críticas, sendo apontado por estes como a solução ideal para o problema. Alguns desses trabalhos serão discutidos mais a fundo no próximo Capítulo.

2.3 Data Diodes

Data Diodes, também conhecidos como *One-Way Data Transfer* [10] ou *Unidirectional Security Gateway* [8], são utilizados para interligar segmentos distintos de rede, permitindo tráfego em apenas uma direção e impossibilitando o retorno de dados na direção oposta [10]. Esses dispositivos, aplicados na integração de redes, se propõem a impedir a passagem de códigos maliciosos para o segmento mais crítico da rede e a invasão e controle dos sistemas e dispositivos internos por entidades não autorizadas.

A principal motivação para implantação de *Data Diodes* consiste na interligação entre segmentos de rede com políticas e níveis de segurança diferentes através de um dispositivo que fisicamente não permita o tráfego bidirecional entre as mesmas [11]. Por isso, podem e vem sendo utilizados para integrar redes de infraestrutura crítica com redes corporativas consideradas inseguras.

Segundo Jeon e Na [4] existem cinco (05) políticas para implantação de *Data Diodes*. São elas:

1. **Comunicação Simples de Caminho Único** - consiste na estrutura mais simples de comunicação, onde o tráfego segue uma única direção, em tempo real, impossibilitando retransmissão e prejudicando a integridade da informação (Figura 2.2).

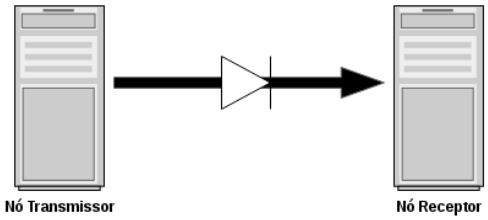

Figura 2.2: Comunicação Simples de Caminho Único. Adaptado de [4]

2. **Comunicação Bilateral usando Múltiplos *Data Diodes*** - utiliza duas comunicações unidirecionais paralelas, em sentidos contrários, entre dois nós de uma rede (Figura 2.3).

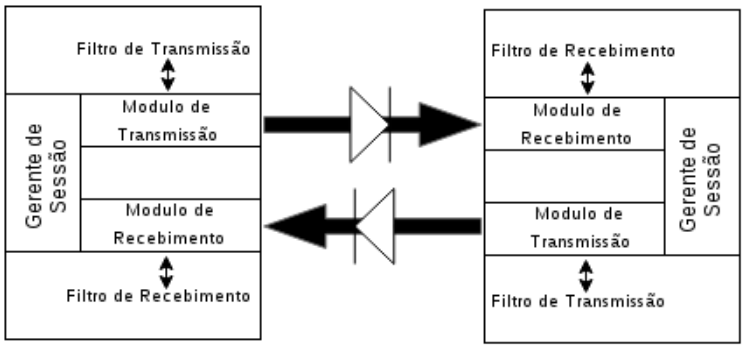

Figura 2.3: Comunicação Bilateral usando Múltiplos *Data Diodes*. Adaptado de [4]

Para o funcionamento dessa política se faz necessário a utilização de uma aplicação de transferência de dados (gerente de sessão) para controlar os diodos e o processo de comunicação. Além disso, também utiliza filtros de envio e recebimento, permitindo apenas que dados pré-definidos sejam transmitidos.

3. **Sinal de Controle de Transmissão por um Nó *Feedack*** - três nós de comunicação (transmissor, receptor e *feedback*) são interligados por *Data Diodes* (Figura 2.4).

   O nó *feedback* recebe a informação sobre a verificação de integridade do nó receptor e quando erros nos dados transmitidos são percebidos, um pedido de retransmissão é enviado pelo nó de *feedback* ao nó transmissor. Esse método apresenta grande vantagem por garantir controle de retransmissão de dados e integridade das informações transferidas.

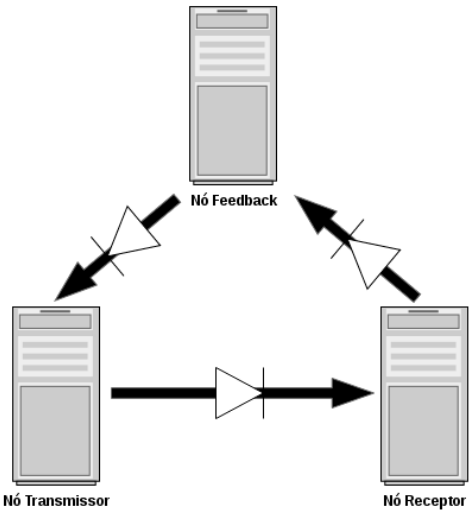

Figura 2.4: Sinal de Controle de Transmissão por um Nó *Feedack*. Adaptado de [4]

4. **Usando o Sinal de Controle de Memória** - possui duas interligações de sentido único entre as redes de comunicação (Figura 2.5).

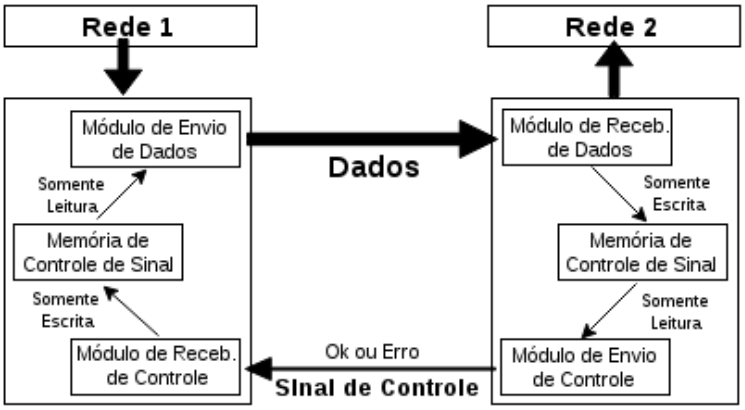

Figura 2.5: Usando o Sinal de Controle de Memória. Adaptado de [4]

A primeira transfere dados no sentido Rede 1 para Rede 2 e a segunda envia sinais de controle entre Rede 2 e Rede 1. Dentro dos dispositivos presentes em cada segmento, ainda existem configurações de que permitem somente a leitura ou escrita de dados em determinadas regiões de memória, a fim de impedir a utilização desse link de controle para outros fins.

5. **Política para Aplicação do *Forward Error Correction*(FEC)** - em

uma transmissão *one-way* simples, não existe retransmissão de dados, portanto essa política utiliza algoritmos de *Forward Error Correction* (FEC) para correção de erros. A técnica FEC é utilizada para controlar erros em comunicações de dados em canais ruidosos ou fisicamente não confiáveis. A ideia central consiste no emissor enviar a mensagem de forma redundante, utilizando um código de correção de erros como o Hamming Code, Turbo Code, Reed-Solomon ou outros.

2.3.1 Desafios com Data Diodes

A operação de um *Data Diode* em ambientes reais apresenta uma série de dificuldades. A primeira é como trafegar uma variedade de protocolos de comunicação bidirecionais por meio de um canal unidirecional [10]. Protocolos baseados em TCP (*Transmission Control Protocol*) não se adequam ao uso em *gateways* unidirecionais por se tratar de um protocolo orientado à conexão. Portanto, o protocolo mais apropriado para uso em *Data Diodes* é o UDP (*User Datagram Protocol*) que não exige estabelecimento de conexão nem troca de dados de controle. Por sua vez, o protocolo UDP não possui nenhum mecanismo que garanta o ordenamento dos pacotes e a integridade dos dados transmitidos.

Outro desafio é a confiabilidade da transmissão unidirecional ser afetada pela falta de um mecanismo eficiente de retransmissão de dados, o que impacta diretamente no desempenho da comunicação [4]. Alguns trabalhos tentam solucionar esse problema através do código Reed-Solomon ou FEC (*Forward Error Correction*), o que permite recuperar partes da informação sem a necessidade de retransmissão [8]. Contudo, o Reed-Solomon ao utilizar n caracteres para verificação torna possível detectar n caracteres incorretos e recuperar apenas $n/2$ caracteres da informação total enviada. Assim, em transmissões ruidosas ou com alto índice de perda de quadros a integridade dos dados estará totalmente comprometida.

O terceiro desafio é que não havendo estabelecimento de conexão entre origem e destino, também não existe relação de confiança na comunicação, ou seja, não há garantia de que a origem é realmente o ativo responsável por transmitir os dados para o outro segmento de rede ou que o destinatário é o sistema correto para receber as informações e processá-las. As propostas e implementações de *Data Diodes* disponíveis na literatura são completamente suscetíveis a ataques *man-in-the-middle*, que podem capturar as informações trafegadas ou mesmo alterar dados em tempo real sem o menor risco de detecção pelos equipamentos envolvidos. Os mecanismos de estabelecimento de conexão e verificação mais simples (ainda que falhos) das comunicações bidirecionais não estão presentes no cenário unidirecional. Portanto, se faz necessária a implementação de um mecanismo seguro de autenticação mútua e garantia de confidencialidade da informação e

confiabilidade entre dispositivos, como, por exemplo, um TPM.

2.4 TPM

TPM (*Trusted Platform Module*) é uma tecnologia de segurança de dados surgida na última década e alvo de vários estudos que buscam comprovar sua efetividade ou encontrar falhas em sua arquitetura [12]. Fisicamente, é um chip composto por um coprocessador criptográfico[1].

A tecnologia TPM possui três importantes funções [13] que garantem a integridade do sistema local e fornecem um mecanismo confiável de autenticação remota totalmente baseado em hardware, o que justificam seu uso neste trabalho. São elas:

1. **Measured Boot**, que verifica a integridade dos arquivos de inicialização através de *hash*;
2. **Sealed Storage**, as chaves ou dados da plataforma ficam "selados" (protegidos) dentro do chip;
3. **Network Attestation**, atestação e autenticação remota, via rede, entre dispositivos.

O **Measured Boot** é um processo no qual o software de pré-inicialização do sistema (*boot*) utiliza o TPM para verificar se não houveram mudanças no ambiente de inicialização através de *hashes* obtidos previamente [13]. Durante a inicialização do sistema, há uma medição de todos os componentes utilizados (BIOS, *bootloader* e sistemas operacionais) gerando um valor de *hash* armazenado no TPM, gerando assim uma cadeia de confiança. Esses valores são registrados pelo TPM em um conjunto de registros chamado PCR (*Platform Configuration Registers*), que não podem ser ajustados ou diretamente introduzidos por nenhum hardware ou software, a não ser o próprio TPM. Os valores registrados no PCR recebem uma assinatura criptográfica com uma chave interna do TPM, que servirá como um atestado da configuração de software do dispositivo, que poderá ser enviada posteriormente a um sistema solicitante como prova do estado e integridade do sistema [12].

Após a configuração correta dos valores no PCR, ocorre o processo **Sealed Storage**, onde as chaves criptográficas disponíveis no dispositivo serão seladas,

[1] Atualmente é fabricado por diversas empresas do segmento de hardware (como a Broadcom, Atmel, Infineon, STMicroeletronics e Nuvoton) e já incluído em alguns computadores e notebooks [13].

se tornando acessíveis apenas se o processo *Measured Boot* numa próxima inicialização encontrar o sistema no mesmo estado específico, gravado anteriormente [13]. Assim, se um malware se instala alterando o processo de inicialização do sistema, as chaves ficarão inacessíveis.

Por fim, **Network Attestation** consiste na prova criptográfica a um host remoto de que a plataforma está em um estado particular (considerado seguro), para que possa se comunicar com outros dispositivos. A Figura 2.6 ilustra o processo.

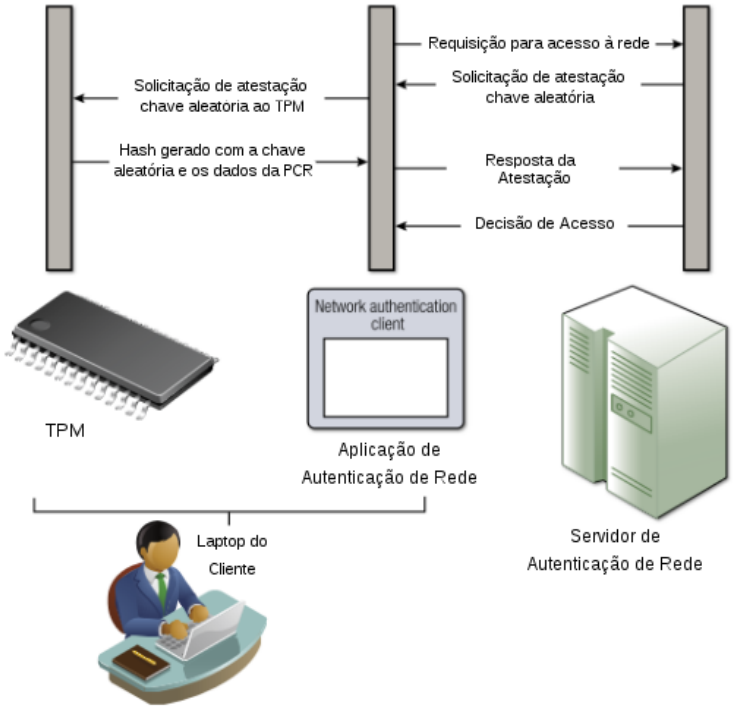

Figura 2.6: Troca de Mensagens do *Network Attestation*. Adaptado de [13]

O servidor cria uma chave criptográfica aleatória e envia ao cliente. Essa chave aleatória é enviada ao TPM, que gera um *hash* com os valores presentes na PCR juntamente com a chave enviada, que é assinado por uma chave de identificação. Esses dados são enviados ao servidor, onde é verificado o estado da plataforma e a chave de identificação para liberação de acesso à rede [13].

2.5 Comunicação UDP

O protocolo TCP é amplamente utilizado em comunicações de dados, sejam através da Internet ou das redes locais. Sua vasta utilização está baseada na con-

fiabilidade, controle de fluxo e controle de congestionamento que o protocolo oferece [14]. Em ICS, as características de segurança do protocolo TCP são imprescindíveis (checagens e garantias de entrega), tendo em vista a necessidade de confiabilidade no processo de comunicação. Porém, os protocolos que funcionam sobre TCP não podem ser implementados em ambientes que utilizem comunicação unidirecional.

Já o protocolo UDP é largamente utilizado em transmissões de dados não sensíveis a atraso e tolerantes à falhas [14], visto que apresenta uma estrutura muito mais simples e de rápido encaminhamento, mas não tem suporte a controles de fluxo e congestionamento ou garantias de comunicação fim-a-fim.

Mas como ou porque usar esse protocolo em redes de comunicação unidirecional? Vários estudos vêm sendo realizados com o objetivo de aplicar ao UDP mecanismos de perda de dados e garantias de comunicação mais confiável, sem alterar suas características. Sneha e Thornbre, em seu artigo *Modelling of UDP Throughput* [14], provaram, através de simulações, que fatores como o tamanho do datagrama, número de saltos e tamanho de *buffer* podem influenciar na diminuição de erros em uma comunicação UDP. Segundo os autores, considerando uma rede onde há apenas tráfego UDP, datagramas de tamanho superior a 210 bytes degradam a transmissão de dados, chegando ao ponto mais alto de degradação em datagramas maiores que 240 bytes. Após os experimentos, eles definiram que o tamanho de datagrama de 210 bytes ou inferior deve ser o padrão para comunicações UDP. Outro ponto observado é que, pela ausência no controle de fluxo, uma transmissão UDP degrada a partir do vigésimo salto, levando a um efeito cumulativo de perda de datagramas.

Já Syed Humair Ali et al. [15] afirmaram que a perda de dados em uma transmissão UDP também está relacionada ao tamanho do *buffer* dos dispositivos envolvidos na transmissão. A perda de dados em uma comunicação UDP tende a cair de acordo com o aumento do *buffer*, ou seja, quanto maior o *buffer* menores serão as perdas. Em comunicações TCP, os autores foram capazes de zerar as perdas de pacotes com a utilização de um *buffer* 50 vezes maior que um pacote transmitido. Já para o tráfego UDP isso não foi possível. Mesmo considerando um cenário de testes que possuía um gargalo de comunicação de dados entre dois segmentos de rede apresentados, houve redução da probabilidade das perdas a partir da utilização de um *buffer* 150 vezes maior que o datagrama UDP.

Com base nos dados apresentados, constata-se a possibilidade de diminuir as perdas em uma comunicação UDP através da diminuição do tamanho do datagrama, diminuição da quantidade de saltos entre origem e destino e aumento do tamanho de *buffer* dos dispositivos envolvidos, tornando assim o tráfego UDP viável para a comunicação entre os dispositivos que possam via a fazer parte de uma comunicação de dados unidirecional.

Capítulo 3

Trabalhos Relacionados

Este Capítulo apresenta os trabalhos relacionados bem como uma discussão sobre suas vantagens e desvantagens. Para tanto, realizou-se uma Revisão Sistemática da Literatura (RSL) sobre os esquemas de segurança da informação aplicados em Redes Industriais e a utilização de *Data Diodes* e TPM. O processo RSL é descrito no Anexo A.

3.1 Trabalhos Correlatos

Os trabalhos correlatos e relacionados à esta dissertação são explicados a seguir.

3.1.1 Secure One-Way Data Transfer

Arkhangelskii et al. [11] desenvolveram um projeto de *Data Diode* provido por um chip FPGA (*Field Programable Gate Array*), que realiza a distinção entre tráfego *stream* (movido diretamente entre as interfaces através do chip FPGA) e arquivos (enviados a uma CPU), guardando as informações em um armazenamento antes de encaminhá-las à interface de saída (Figura 3.1).

De acordo com a proposta, em casos de erro de transmissão os dados são lidos a partir do armazenamento e reenviados. Para transportar os dados, foi implementado o protocolo UDP com algumas alterações, como um registro de sequência de pacotes, campos de informação e campos de confirmação.

Os experimentos foram realizados em um computador com processador Core i7, 16Gb de memória RAM, disco SSD de 1Tb e interfaces de 1Gbps. Para tráfego *stream*, a velocidade média alcançada foi de 100Mbps, com perda de dados variando de 0,05%, para dados de 1GB, e 0,14% para dados de 10GB. Já no tráfego de arquivos, a velocidade média de transmissão foi cerca de 60Mbps,

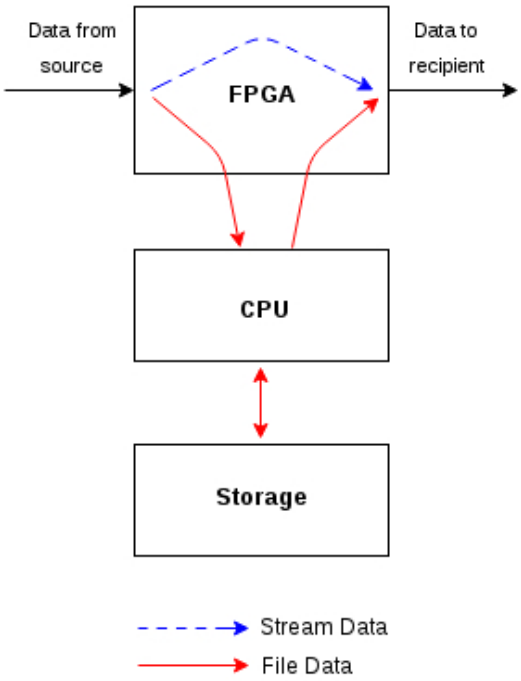

Figura 3.1: Arquitetura de *Data Diode* proposta por Arkhangelskii et al. Adaptado de [11]

com uma média de retransmissão em 4%, não chegando a ultrapassar a barreira dos 4,2%.

A principal contribuição do trabalho está no fato de resolver o problema da retransmissão de dados em arquiteturas com *Data Diodes*. Porém, o artigo utiliza apenas uma versão customizada do protocolo UDP, o que dificulta a integração com dispositivos já existentes no mercado. Além disso, os testes não foram muito bem documentados, o que torna difícil sua reprodução em outros ambientes e arquiteturas para comparação de resultados. Por fim, não são abordados os aspectos de Segurança da Informação no projeto.

3.1.2 UNIWAY

Heo et al. [8], em seu artigo " Design of Unidirectional Security Gateway for Enforcement Reliability and Security of Transmission Data in Industrial Control Systems", propuseram um sistema de *gateway* unidirecional, chamado UNIWAY, que contém dois dispositivos interligados por um meio físico, permitindo comunicação em apenas uma direção e comunicação bidirecional apenas nas extremidades do conjunto. Funcionando como um *Simple One-way Communication* melhorado, é dividido em três módulos (*Transfer Application Proxy*, *Transfer*

Data Manager e *Transfer NIC Card*) e realiza tarefas como filtragem de endereço IP e Porta, filtragem de conteúdo e sequenciador de dados, conforme Figura 3.2.

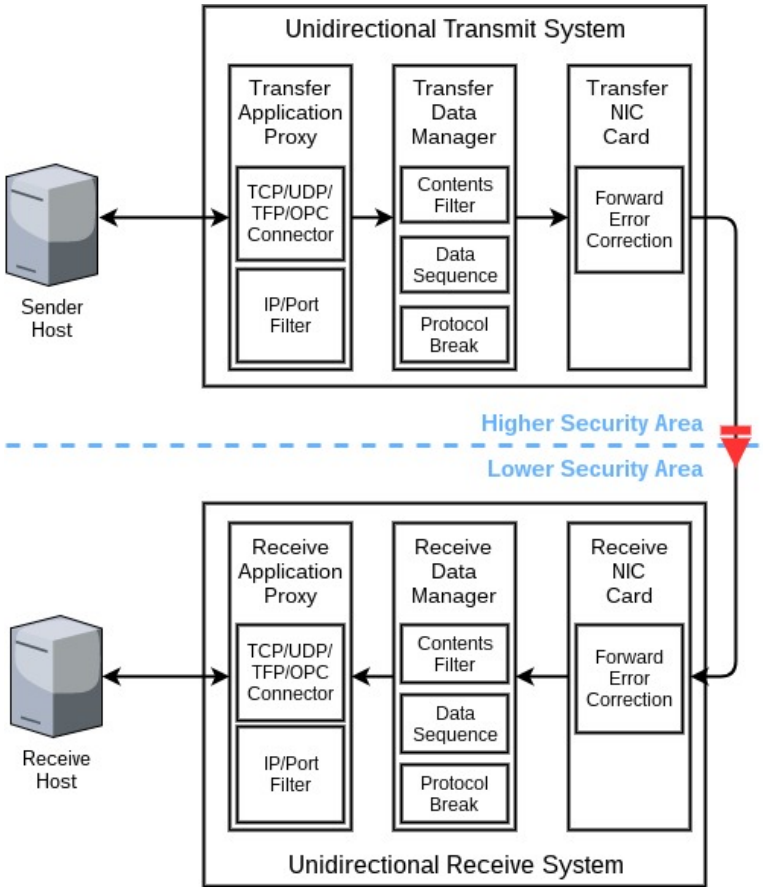

Figura 3.2: Arquitetura do UNIWAY. Adaptado de [8]

Para a comunicação entre as redes onde se encontram os dispositivos de origem e destino (em cada uma das extremidades do UNIWAY) utilizam-se proxyes TCP, UDP e de aplicação, estabelecendo um esquema de comunicação tradicional com os hosts (como por exemplo o *three way handshake* TCP). A partir da entrada dos dados no dispositivo é realizada filtragem de IP/Porta e conteúdo, a fim de garantir a integridade dos mesmos. Posteriormente esses pacotes são submetidos a um sequenciador de dados e um *protocol break*, que organiza os dados em um novo formato constituído dos seguintes campos: ID de mensagem, número de sequência, tamanho e dados. Por fim, a mensagem é enviada à interface de transferência, onde ainda é submetida a um tratamento pelo algoritmo Reed-Solomon para uma possível necessidade de reparação de dados no outro lado do

diodo. Chegando a outra extremidade, o dado passa pelo processo inverso ao anterior para se obter o dado transmitido inicialmente. A informação é enviado ao destinatário através de um outro proxy.

O trabalho não apresenta resultados, tendo em vista que apenas a proposição de um esquema, não deixando claro se existem implementação. No entanto, não é tratado o problema da retransmissão, tendo em vista que o algoritmo Reed-Solomon consegue recuperar apenas uma parte da informação. Além disso, por utilizar proxies TCP não é possível garantir que origem e destino sejam íntegros, confiáveis e autênticos.

3.1.3 Development of unidirectional security gateway appliance using intel 82580EB NIC interface

Heo et al. [16] implementaram sua proposta de UNIWAY em uma arquitetura Intel 82580EB. O artigo valida a arquitetura proposta no trabalho anterior [8], implementando através de um hardware Intel seu *gateway* unidirecional. A implementação utiliza dois equipamentos para compor o diodo, interligados via cabo de fibra ótica. A comunicação do cliente com o Diodo é TCP, convertida para UDP e transferida unidirecionalmente para a segunda unidade, que entrega os dados ainda via UDP a um destino final.

Os testes não foram muito bem detalhados, tendo em vista que apenas os tamanhos de pacotes e as velocidades de transmissão foram medidas, faltando métricas e análises importantes, como, por exemplo, a perda de dados e o comportamento frente aos ataques possíveis à solução. Por conta desses fatores, não é possível comparar os resultados obtidos nesse trabalho com os dados a serem medidos nesta dissertação.

3.1.4 A Study on the Communication Agent Model for One-way Data Transfer System

Oh et al. [10] usa as falhas de segurança e vulnerabilidades das redes industriais modernas como plano de fundo para propor um modelo de comunicação utilizando *Data Diodes* para interligar o segmento da rede de controle com a rede de serviços. O artigo se propõe a transmitir dados de protocolos baseados em TCP/IP por um gateway unidirecional sem alterar o estado dos pacotes e da conexão, mas transferindo tudo (dados e cabeçalhos originais) ao seu destino final.

O modelo é composto por:

1. Dispositivo de Link - trabalha com interfaces de Fibra Ótica, onde no transmissor o RX de uma das interfaces é conectado no TX da interface secun-

dária do próprio equipamento e o TX é interligado diretamente no RX da placa do segundo equipamento;

2. Servidor de Transmissão - se comunica com a rede de controle e armazena todos os dados trocados nessa sessão, inclusive os cabeçalhos e dados de controle. Transmite de forma unidirecional ao Servidor de Recepção todos os dados capturados;

3. Servidor de Recepção - recebe os dados capturados pelo Servidor de Transmissão e encaminha esses dados ao Agente de Serviço;

4. Agente de Serviço - recebe múltiplas sessões de transmissão e fica responsável por enviar os dados aos clientes corretos de cada sessão, opera um canal de multi sessão N:N onde trata os dados que devem ser encaminhados para cada um dos destinos correspondentes;

Os autores realizaram testes com a transmissão de aproximadamente 100GB de dados capturados através do software TCPdump, mas não foram realizados experimentos com comunicação de rede em um ambiente de infraestrutura segmentada. Foram simuladas as transferências de dados a partir do arquivo .pcap, as quais apresentaram 0% de perda no ambiente proposto. Os testes não são bem explicados e nem o hardware implementado foi apresentado, havendo apenas informações muito superficiais.

3.1.5 Unidirectional Secure Information Transfer via RabbitMQ

Melorose et al. [6] implementa um barramento de mensagens que possa ser utilizado de maneira transparente sobre um *Data Diode*, utilizando a aplicação de código aberto RabbitMQ, que é um software de mensagens *open-source* que suporta o protocolo AMQP (*Advanced Message Queuing Protocol*). Mesmo o protocolo sendo originalmente utilizado para o transporte de mensagens, foram utilizados diferentes tipos de dados, inclusive informações criptografadas.

A arquitetura da solução proposta por Melorose et al. contém os seguintes itens (conforme imagem 3.3):

1. *Encrypt*: Responsável por encriptar todos os dados que chegam ao Diodo;

2. *Cutter*: Realiza a quebra dos dados em segmentos de 8KB, além de enumerá-los para posterior ordenação e duplicá-los antes do envio;

3. *UDP Sender*: Envia os dados ao receptor do outro lado do Diodo;

4. *UDP Receiver*: Recebe os dados enviados por meio do Diodo;
5. *Merger*: Deduplica, ordena e monta os dados recebidos para que se tornem de acordo com a mensagem anteriormente enviada;
6. *Decrypt*: Descriptografa os dados e gera novamente a informação original.

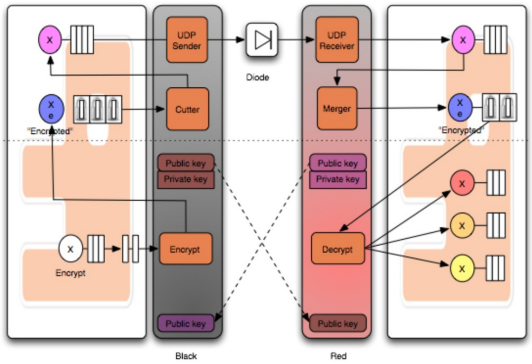

Figura 3.3: Arquitetura proposta por Melorose et al. Disponível em [6]

A implementação realizada com os componentes acima listados funcionando sobre o software RabbitMQ chegou a um resultado de transmissão de dados a uma velocidade de 25 MBps em uma conexão Gigabit sem ocorrência de perda de dados. Todos os dados foram transmitidos em duplicidade, com o objetivo de diminuir ao máximo as perdas de dados, o que acaba impactando no throughput dos dados a serem transmitidos.

A criptografia de dados entre os componentes do Diodo nessa implementação é válido, pois os autores deixam em aberto a forma de interligação entre os dois dispositivos encontrados nos extremos da comunicação unidirecional. Porém seria útil que os dados fossem criptografados até o destino final, principalmente levando-se em consideração que a rede de destino possa ser tão ou até mesmo mais vulnerável que a rede de comunicação entre os componentes do Diodo.

3.1.6 Data diodes in support of trustworthy cyber infrastructure

Okhravi e Sheldon [9] descrevem nesse trabalho riscos relacionados à Segurança da Informação em um ambiente de PCN (*Process Control Network*). Segundo os autores, falha na configuração de *Firewall's*, dispositivos vulneráveis, falhas de software e acesso físico inseguro à parte da rede são graves ameaças a sistemas críticos, representando problemas na integração de redes industriais.

Como resposta aos problemas apontados, os autores descrevem a arquitetura TPCN (*Trusted Process Control Network* com o uso de *Data Diodes*. Essa arquitetura se assemelha aos outros trabalhos citados contendo *Data Diodes*, porém possui um dispositivo chamado *Authentication, Authorization, and Access Control (AAA) Server*, acessível pelas duas redes interligadas, que controla as políticas de acesso com base na avaliação de requisitos pré definidos (*compliance*) nos dispositivos envolvidos. Somente dispositivos de acordo com as políticas de acesso podem enviar dados entre a rede PCN e a rede corporativa, bem como recebê-los.

Esse trabalho é o primeiro a contar com uma avaliação de segurança nos dispositivos envolvidos na comunicação, porém o fato de existir um servidor que detenha comunicação bidirecional com as duas redes, acaba por comprometer o uso do diodo e a segurança física provida pela comunicação unidirecional. O fato de existir um servidor com comunicação fisicamente bidirecional entre as redes, faz com que o elo mais fraco nessa comunicação seja justamente este equipamento, à partir do seu comprometimento um atacante poderia escalar para qualquer uma das redes. A ideia de validar os dispositivos é muito interessante, porém deve ser implementada de maneira diferente, a fim de não criar um caminho bidirecional entre os dois lados do diodo.

3.2 Discussão

Embora os trabalhos apresentados lidem com o conceito de *Data Diodes*, propondo, implementando e testando arquiteturas, um grande problema encontrado na comunicação unidirecional consiste na impossibilidade de retransmissão oferecida tradicionalmente pelo protocolo TCP. De acordo com o ambiente de rede utilizado e a quantidade de ruído gerada no meio físico, a taxa de pacotes perdidos pode ser extremamente alta, impossibilitando uma transferência de dados eficiente e impactando diretamente na integridade das informações no destino.

Além disso, a não utilização dos protocolos tradicionais de transmissão de dados, impacta diretamente na integridade das informações transmitidas entre segmentos e na arquitetura de comunicação proposta pelos fabricantes. Havendo a necessidade de implantar novos protocolos, corre-se o risco de se descartar sistemas legados, proporcionando um grande gasto para as indústrias e a padronização todos os dispositivos para atuarem com esse novo protocolo seria extremamente demorada, visto que uma ação como essa durou vários anos para que fosse completamente realizada.

Esses graves problemas acabam por inviabilizar a implantação de *Data Diodes* em redes industriais, deixando essas infraestruturas desprotegidas e suscetíveis a todo tipo de ameaça.

Capítulo 4

Proposta

Este Capítulo apresenta o *Data Diode* proposto, descrevendo um modelo funcional composto por elementos (*building blocks*) que possibilita diferentes formas de implementação. No quesito segurança, ele difere-se das soluções existentes na literatura ao prover a verificação da integridade dos participantes da comunicação, através da atestação remota e autenticação mútua das partes, permitindo o estabelecimento de relação de confiança entre os dispositivos envolvidos. Além disso, garante confiabilidade na comunicação (retransmissão de dados).

4.1 Modelo Funcional

A proposta de *Data Diode* seguro e confiável é baseada em componentes e artefatos, conforme representado na Figura 4.1.

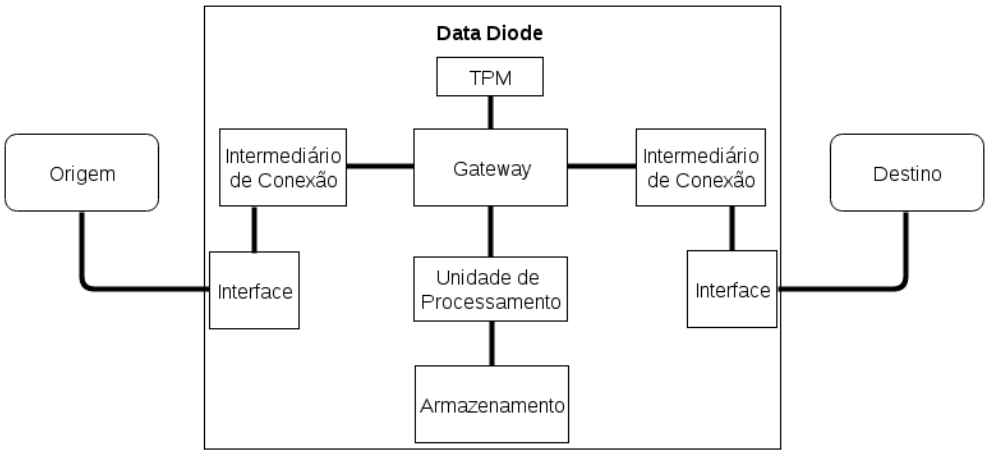

Figura 4.1: Arquitetura Proposta do *Data Diode*

Os seis elementos principais do *Data Diode* proposto são: **Interfaces de Conexão**, **Intermediário de Conexão**, **Gateway**, **Unidade de Processamento**, **Armazenamento e Unidade TPM**. É importante destacar que os dispositivos de origem e destino dos dados também possuem como requisito estrutural para seu funcionamento uma unidade TPM para atender requisitos de atestação e autenticação no dispositivo principal.

As **Interfaces de Conexão** fazem a ligação da origem e destino com o *Data Diode*, permitindo a entrada de dados por parte da origem e a entrega de dados para o destino. Também gerenciam o controle de fluxo de comunicação e *buffers* para transferência de dados.

Juntamente com as Interfaces de Conexão funcionam os **Intermediários de Conexão**, os responsáveis pela interação TCP/UDP com os dispositivos de origem e destino da comunicação. Em outras palavras, o estabelecimento de toda e qualquer conexão entre origem e destino é, de fato, realizada entre a origem e o Intermediário de Conexão, e, posteriormente, entre o Intermediário de Conexão e o destino. Todos os procedimentos utilizados em uma conexão TCP (*three-way-handshake*, por exemplo) são realizados pelo Intermediário, garantindo retransmissão e negociação na troca de dados.

No **Gateway** é implementada a comunicação unidirecional, impossibilitando o tráfego na direção contrária à estabelecida. Fugindo um pouco do conceito da palavra *Gateway*, esse componente não permite a passagem de dados no sentido destino-origem. Também impossibilita a gravação direta de dados no Armazenamento pelo destino, permitindo apenas a leitura, visto que toda a comunicação entre interfaces e Unidade de Processamento/Armazenamento é gerenciada pelo Gateway.

A **Unidade de Processamento** coordena todo processo de escrita/leitura na região da Unidade de Armazenamento destinada à guarda dos dados. Funções de sequenciamento dos dados e controle de espaço ocupado são implementadas como tarefas da Unidade de Armazenamento.

Na **Unidade de Armazenamento** são gravados todos os dados trocados entre origem e destino. Havendo necessidade, o destino pode solicitar a retransmissão ao *Data Diode* e o dado será reenviado a partir das informações armazenadas. Essa unidade também contém os registros (*logs*) de transmissão de dados e arquivos necessários ao funcionamento do *Data Diode*, obviamente em regiões separadas logicamente.

O **TPM** é o responsável por implementar segurança no dispositivo através de funções de verificação de integridade, atestação remota e autenticação de dispositivos. A verificação de integridade ocorre na inicialização do sistema, onde é verificado o estado dos dados do sistema presentes no armazenamento, a fim de detectar possíveis anomalias ou códigos maliciosos que possam ter se instalado

no dispositivo. A partir da validação do *boot*, iniciam-se as fases de atestação remota, para garantir aos dispositivos remotos a integridade do *Data Diode*, e autenticação mútua, para garantir que somente dispositivos autorizados enviem ou recebam informações.

4.2 Etapas de Comunicação

Antes da transferência de dados entre origem e destino, as garantias de integridade dos dados, confiabilidade entre os dispositivos e autenticação precisam ser estabelecidas, objetivando a segurança dos dados a serem trafegados. Para isso três etapas ocorrem envolvendo o *Data Diode* e os dispositivos conectados a ele nas duas redes.

A primeira etapa é o *Measured Boot*, que mede os valores de inicialização do dispositivo, para comparar esses valores adquiridos com os valores armazenados da última inicialização considerada segura. Ela ocorre em todos os dispositivos envolvidos (Figura 4.2) durante o processo de inicialização do sistema.

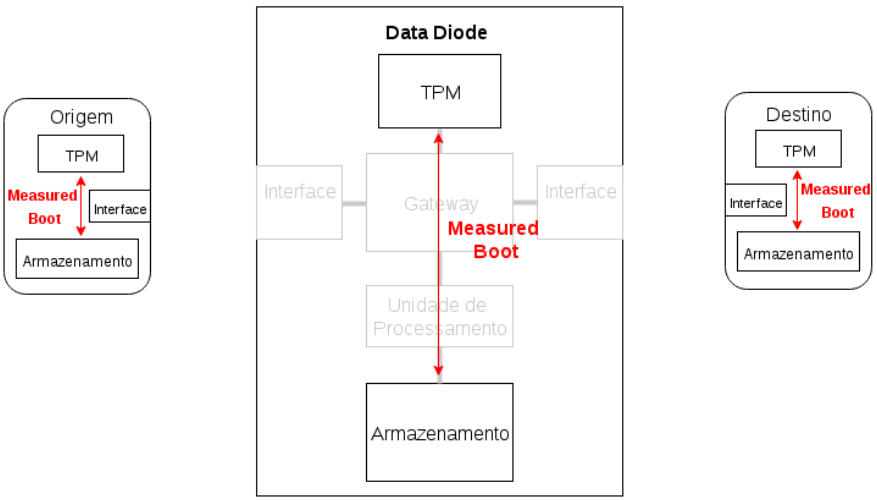

Figura 4.2: *Measured Boot*

Essa etapa não precisa ser sincronizada entre os dispositivos, porém todos precisam passar por ela antes do início da troca de dados. A cada reinicialização do sistema, o *Measured Boot* é executado e o funcionamento dos dispositivos depende diretamente da integridade do processo de inicialização. No caso da mensuração apresentar um valor insatisfatório por conta de alterações indevidas na inicialização ou arquivos do sistema, o dispositivo em questão não poderá avançar para a próxima etapa, ficando assim incomunicável, pois as chaves necessárias para o *Network Attestation* estarão seladas (por conta do processo de

Sealed Storage), impossibilitando seu acesso.

A etapa seguinte (segunda) é o processo *Network Attestation*, onde os dispositivos atestam remotamente a sua integridade, oferecendo uma garantia baseada em hardware e se autenticam mutuamente (Figura 4.3).

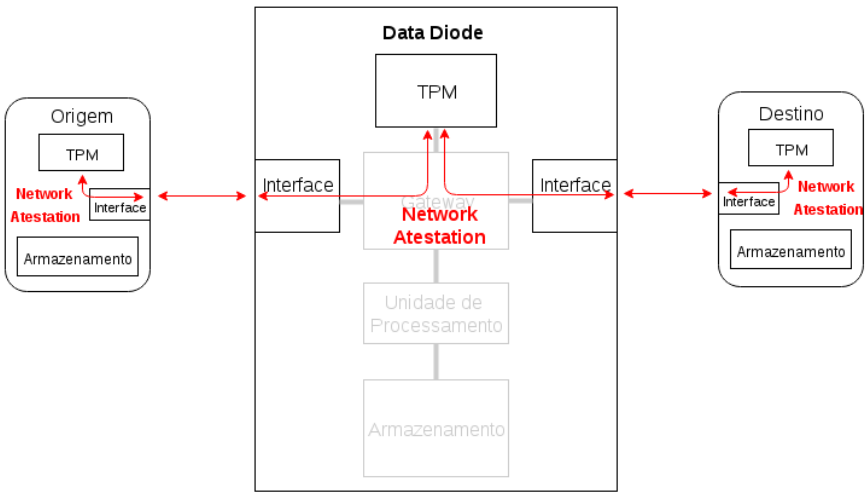

Figura 4.3: *Network Attestation*

A atestação ocorre entre o *Data Diode* e os dispositivos de origem e destino dos dados separadamente, ou seja, são dois processos que ocorrem simultaneamente. Não há atestação direta entre origem e destino, porém o *Data Diode* é o servidor de autenticação e realiza o processo nas duas direções, assim garantindo a integridade de todos os dispositivos envolvidos.

Por fim, a última etapa é a transferência de dados (Figura 4.4).

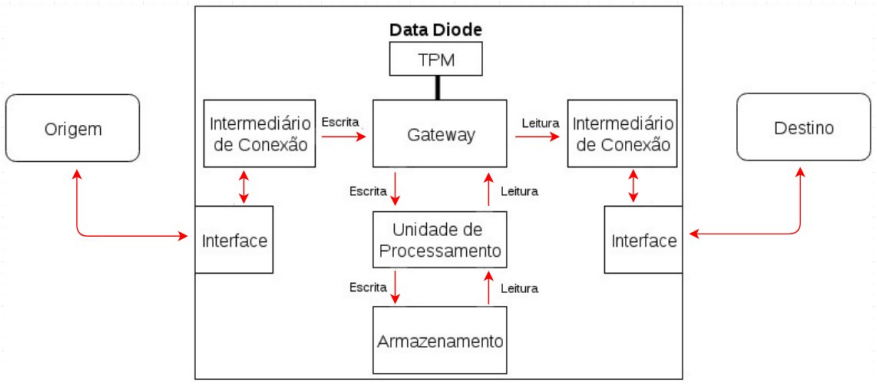

Figura 4.4: Processo de Comunicação

O dispositivo de origem transfere dados através de protocolo pré-determinado com o *Data Diode*, que realiza todas as interações como sendo o destino final da

conexão através do Intermediário de Conexão. No caso de protocolos baseados em TCP, todo estabelecimento e negociação dos parâmetros de conexão é realizado entre origem e Intermediário de Conexão. A partir do recebimento de dados, o *Data Diode* armazena essas informações, descartando informações desnecessárias e pertinentes ao estabelecimento de conexões anteriores. As informações armazenadas recebem um número sequenciamento e dados de sessão, para evitar que dados de conexões diferentes possam se misturar em meio ao processo de armazenamento.

Em seguida são enviadas pelo próximo Intermediário de Conexão, que estabelece conexão e define os parâmetros de comunicação com o destino (no caso de conexões TCP) e envia os dados recebidos pela conexão de entrada anteriormente. Caso o destino detecte erros nos dados recebidos, poderá solicitar a retransmissão dos mesmos para o Intermediário de Comunicação, que reenviará as informações através dos dados armazenados anteriormente.

4.3 Formas de Implementação

Graças a separação dos elementos funcionais, o *Data Diode* proposto pode ser implementado de diferentes formas sem perder suas principais características e apresentando resultados satisfatórios na comunicação unidirecional. Algumas formas viáveis são descritas a seguir:

4.3.1 FPGA

O *Data Diode* proposto é implementável a partir de um chip FPGA (*Field Programmable Gate Array*), que possibilita a configuração de funções lógicas, afetando diretamente o hardware em questão. Na FPGA pode ser programada a comunicação unidirecional, impossibilitando o tráfego na direção contrária à estabelecida. Toda a comunicação entre interfaces e CPU/Armazenamento é gerenciada pelo chip, fazendo o controle das operações de gravação e leitura que por ventura ocorram. Além disso, a FPGA também não permitirá a passagem de dados no sentido destino-origem, impossibilitando a gravação direta de dados no Armazenamento pelo destino (permitindo apenas a leitura pelo mesmo).

As Interfaces fazem a interconexão da origem e destino com o *Data Diode*, permitindo a entrada de dados por parte da origem e a entrega de dados para o destino. Porém, juntamente com as Interfaces estarão funcionando dois *proxies* de aplicação, que serão responsáveis pela iteração TCP/UDP com os dispositivos de origem e destino de comunicação. Em outras palavras, o estabelecimento de conexão será realizado entre origem e o primeiro Proxy, e posteriormente entre o segundo Proxy e destino. Todos os procedimentos utilizados em uma conexão

TCP serão realizados pelos *proxies*, garantindo retransmissão e negociação na troca de dados.

A Figura 4.5 demonstra um modelo do hardware dessa implementação.

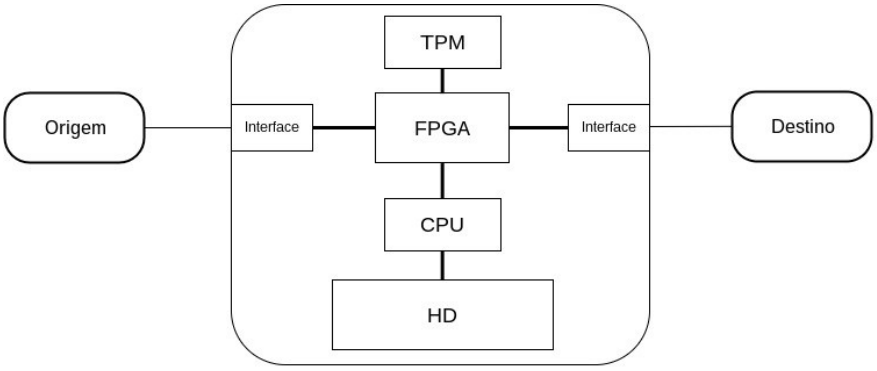

Figura 4.5: Implementação com FPGA

4.3.2 Par de Computadores

Uma segunda forma de implementação possível é a partir da composição do *Data Diode* com dois computadores. Para tanto, ambos devem estar munidos de chip TPM, interfaces de comunicação UTP para transmissões bidirecionais com os dois lados do diodo e uma interface de comunicação unidirecional em cada. Pode-se usar um fibra ótica, como no trabalho de Oh et al. [10].

Nesta forma de implementação, as interfaces de rede estariam conectadas fisicamente em um sentido único, o que impossibilita o uso do protocolo TCP para a comunicação entre os computadores que fazem parte do diodo. Portanto, considerando a existência de sistemas legados que trabalhem unicamente sobre o protocolo TCP, toda comunicação entre as duas redes deverá sofrer a alteração do protocolo da camada de transporte de TCP para UDP e posteriormente, no segundo computador, de UDP para TCP, a fim de ser entregue ao destinatário da comunicação. Requisitos de sequenciamento de datagramas, bufferização e o bom funcionamento do meio físico utilizado são fatores importantes nessa implementação, tendo em vista que uma falha em qualquer um desses itens causaria prejuízos na comunicação.

O armazenamento de dados nesse cenário deve ocorrer no segundo computador (conectado à rede insegura), com o objetivo de assegurar que os dados tenham efetivamente sido transmitidos. Além de funcionar como um *buffer* de armazenamento das mensagens que ainda serão enviadas.

A arquitetura é apresentada na Figura 4.6.

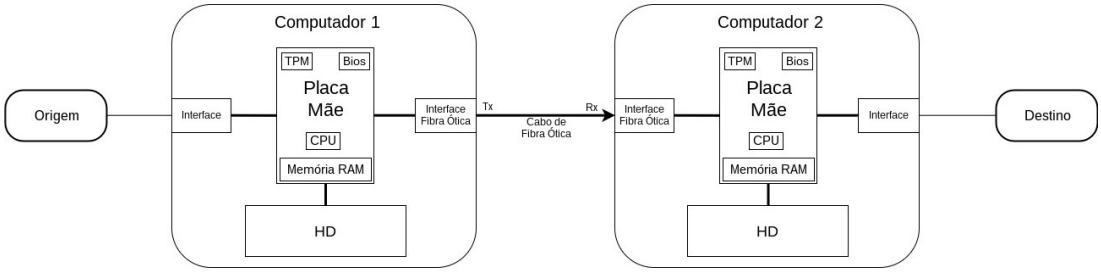

Figura 4.6: Implementação com Par de Computadores

4.3.3 Par de Mini Computadores de Placa Única

Esse modelo de implementação é bem parecido com o anterior, porém existe apenas uma interface ethernet de rede cabeada nas placas utilizada para a comunicação bidirecional com o host de cada uma das redes. Nesse caso, a transmissão unidirecional pode ser obtida através de uma comunicação serial utilizando a interface UART (*Universal Asynchronous Receiver/Transmitter* encontrada no GPIO (*General Purpose Input/Output*) presente nessas placas. Considerando a velocidade mais baixa nesse tipo de comunicação, se torna importante a bufferização dos dados antes do envio para que não haja perda de informação.

O TPM nesses dispositivos pode ser implementado através de uma placa adicional, costumeiramente conhecidas como *shield* ou mesmo através da emulação do chip TPM. Deve-se considerar que mesmo utilizando o *shield* com o chip TPM integrado, pelo fato desses dispositivos não possuírem BIOS, faz-se necessária a implementação das funções de medição de *boot* diretamente no software de inicialização do sistema.

O armazenamento no segundo dispositivo deve utilizar um hardware adicional, tendo em vista o pouco espaço de armazenamento desses dispositivos (geralmente um cartão de memória). Vale a pena frisar que esses dados devem ser armazenados de maneira segura, preferencialmente criptografados com base em chaves contidas no TPM, para se evitar possíveis vazamentos de informação decorrentes de um acesso não autorizado.

A Figura 4.7 demonstra o cenário de implementação dessa arquitetura.

4.4 Modelo de Ameaças

Esta seção apresenta a modelagem de ameaças do *Data Diode* proposto e suas respectivas suposições. São discutidos o modelo de sincronismo e as diferentes ameaças, de forma a permitir que o diodo proposto seja implantável e operacional.

Em primeiro lugar, é assumido um modelo de sincronia parcial, ou seja, o

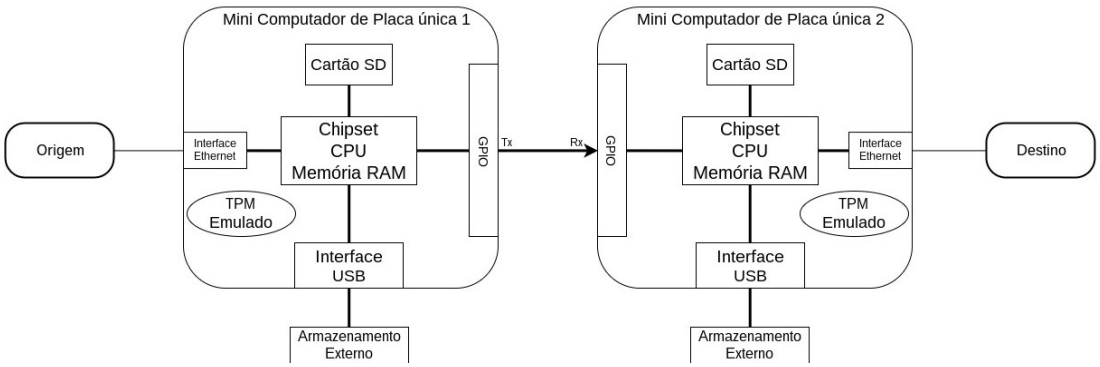

Figura 4.7: Implementação com Par de Mini Computadores de Placa Única

sistema pode se comportar de forma assíncrona por algum tempo, até que se torne síncrono, com limites de tempo de processamento e comunicação. Este é um requisito essencial para garantir o término do processo de inicialização confiável, um alicerce fundamental do *Data Diode* proposto.

Em relação às ameaças, assume-se que as políticas de acesso físico da rede crítica não permitem que um atacante seja capaz de ter o controle de qualquer elemento. Porém, os elementos da rede externa não possuem garantias de acesso, sendo os mesmos acessíveis através da Internet e, portanto, suscetíveis a ataques. Também considera-se um cenário de ameaça onde o atacante tem por objetivo: (i) o controle dos canais de comunicação entre origem e destino e o acesso à rede crítica é feito a partir da Internet; (ii) a obtenção das informações trafegadas entre os elementos de rede presentes.

Entre as ameaças possíveis nesse ambiente estão: *Man-in-the-middle*, Ataques de Replay e Força Bruta. *Man-in-the-middle* é um ataque que consiste na interceptação da comunicação entre dois ou mais dispositivos em uma rede. A técnica mais utilizada para esse tipo de ameaça se dá no envenenamento da tabela ARP dos dispositivos conectados em uma mesma rede, fazendo-os direcionar seu tráfego a um atacante no lugar de um destino válido. Dentro do cenário proposto, esse ataque pode surtir efeito na interceptação da comunicação do diodo com o dispositivo presente na rede insegura, fazendo com que a confidencialidade dos dados transmitidos seja violada.

Ataques de Replay são ameaças que possibilitam um atacante a reutilizar um pacote de autorização antes trafegado na rede, com o objetivo de adquirir uma autenticação frente a um sistema. Considerando a utilização do TPM, um atacante poderia utilizar um dado de atestação remota, trafegado entre o dispositivo presente na rede insegura e o Diodo para se atestar como um dispositivo válido na rede, ou mesmo simular ser o destino final da comunicação devidamente atestado. Esse dado trocado durante o processo de atestação pode também servir

como uma chave que sempre retorne a um servidor de atestação em uma rede, uma medição de boot confiável, mesmo que tenham havido alterações durante o processo.

O ataque de Força Bruta consiste em inúmeras tentativas de autenticação em um dispositivo baseada em uma lista de senhas e/ou nomes de usuários. A partir da rede interna o Diodo pode ser atacado para se tentar uma conexão remota não autorizada.

Capítulo 5

Prototipação e Testes

Este Capítulo tem por objetivo apresentar a implementação do protótipo do *Data Diode* proposto. Para tanto, testes de funcionalidades e verificação de requisitos foram executados e descritos como forma de comprovar a eficácia, em termos de segurança, da solução proposta.

5.1 Prototipação

Para prototipação foi utilizado o modelo de implementação **Par de Mini Computadores de Placa Única**, descrito no Capítulo anterior, devido sua portabilidade e facilidade para a implementação do *Data Diode*.

O hardware utilizado foi um Raspberry Pi 1 modelo B+, que possui uma interface ethernet, quatro interfaces USB 2.0, uma interface HDMI, saída de áudio, uma entrada para cartão Micro SD - funciona como unidade de armazenamento principal do dispositivo - e uma interface GPIO de 40 pinos para a comunicação serial. O processador é um BCM 2835 de 700 MHz, arquitetura 32 bits, com 512 MBytes de memória RAM, que em conjunto representam a Unidade de Processamento descrita no Modelo Funcional. O sistema operacional utilizado foi o Raspbian, uma versão baseada no sistema Debian GNU/Linux Jessie, com kernel 4.9 e lançado em 5 de julho de 2017.

Os computadores dos ambientes da rede crítica e da rede insegura foram implementados em máquinas virtuais, utilizando o sistema operacional Debian Jessie, com os serviços de rede funcionando sobre o protocolo TCP para comunicação entre ambas, sempre no sentido rede crítica para rede insegura. Toda comunicação entre os computadores dos dois ambientes com o *Data Diode* (composto pelos dois dispositivos Raspberry Pi) funciona de maneira bidirecional.

As funcionalidades do chip TPM foram emuladas via software, tendo sido utilizado o *TPM-Emulator* versão 0.7 que emula a especificação 1.2 do TPM

(representando a unidade TPM apresentada no Modelo Funcional). Vale ressaltar que desde 2014 o TCG disponibilizou a versão 2.0 do TPM, porém, por conta das novas funcionalidades e recursos adicionais, essa plataforma ainda não foi emulada via software [17] para a plataforma utilizada, fazendo com que a versão 1.2 seja a utilizada aqui. O software *TrouSerS* foi utilizado em conjunto com o *TPM-Emulator* com o objetivo de acessar as funções disponíveis no TPM emulado.

5.1.1 Data Diode

O *Data Diode* proposto foi implementado através da utilização de dois dispositivos Raspberry Pi, interligados entre si através da interface serial UART presente nos pinos GPIO de ambos, conforme a Figura 5.1.

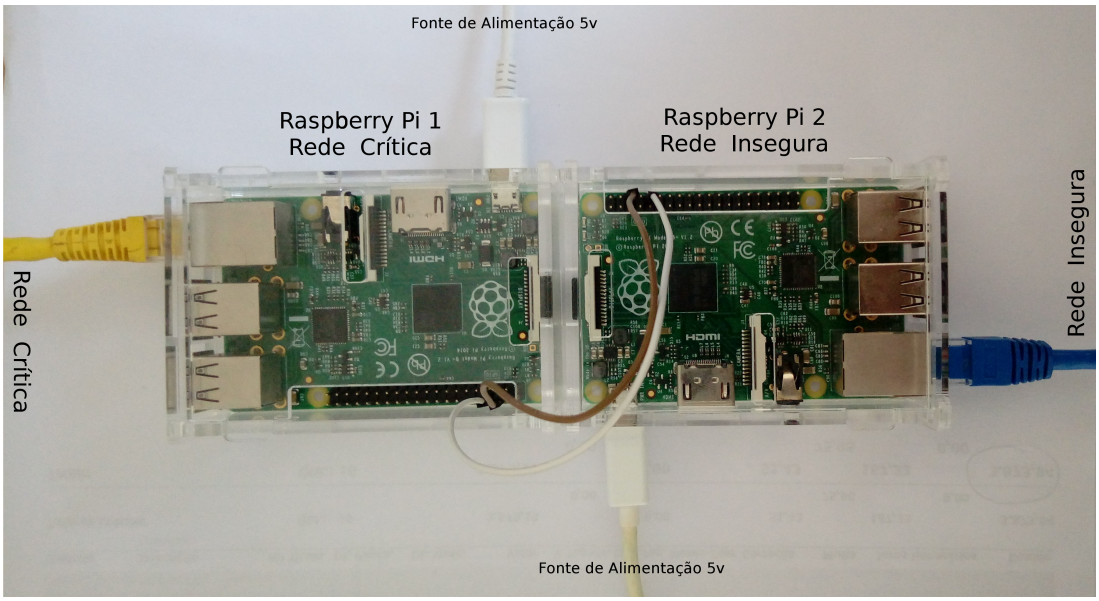

Figura 5.1: Interligação Física do *Data Diode*

Comunicação Unidirecional

Para cumprir o requisito de prover uma comunicação fisicamente unidirecional, foi utilizada a conexão do pino TX (transmissor) do primeiro dispositivo diretamente no pino RX (receptor) do segundo dispositivo, o que restringe o tráfego de informações no sentido contrário ao estabelecido.

Na interface GPIO do Raspberry Pi modelo B+, os pinos 8 e 10 são os responsáveis pelo TX e RX da interface UART respectivamente, conforme mostrado na Figura 5.2. Essa interligação compõe o **Gateway** descrito no Modelo Funcional.

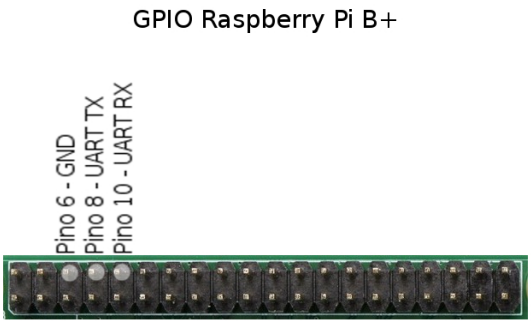

Figura 5.2: GPIO Raspberry Pi B+

Para sua interligação segura, os dois dispositivos devem estar conectados também em uma das interfaces GND (*Graduated Neutral Density*) que funcionam como terra. O pino 6 é um GND e, por sua proximidade com a interface UART, pode ser utilizado nessa interligação. Desta forma, no protótipo foram interligados os pinos 6 dos dois Raspberry Pi B+ e o pino 8 do dispositivo conectado à rede crítica ao pino 10 do dispositivo conectado à rede insegura. A conexão pode ser feita diretamente, sem o uso de uma *protoboard* ou resistores, por conta da compatibilidade eletrônica do dispositivo. Apenas dois cabos foram utilizados para interconexão entre os dispositivos do protótipo (Figura 5.1).

Comunicação com as Redes Crítica e Insegura

No que diz respeito à comunicação do *Data Diode* com as redes crítica e insegura (origem e destino), a interface serial no sistema Raspbian foi implementada com o nome *ppp*0 e configurada com IPv4 na faixa 10.1.1.0/30, sendo 10.1.1.1/30 o transmissor e 10.1.1.2/30 o receptor, conforme Figura 5.3. Considerando que a interface UART funciona de forma assíncrona, em nenhum momento se faz necessária a interligação em sentido contrário para estabelecimento de conexão e sincronismo entre os dispositivos. A partir da configuração das interfaces e da interconexão física entre as mesmas, o sistema operacional já possui condições de trafegar dados unidirecionalmente.

O computador transmissor de dados, localizado na rede crítica, foi interligado ao primeiro Raspberry em sua interface ethernet RJ45, em modo bidirecional e velocidade máxima de 100 Mbps. Da mesma forma o computador receptor, localizado na rede insegura, foi interligado ao segundo Raspberry. Essas conexões com os dispositivos presentes nas redes são representadas pelo componente **Interfaces de Conexão** do Modelo Funcional. As redes crítica e insegura possuem faixas de rede IP (versão 4) distintas, sendo 192.168.0.0/24 para a rede crítica e 192.168.1.0/24 para a rede insegura, conforme Figura 5.3. Estas redes não

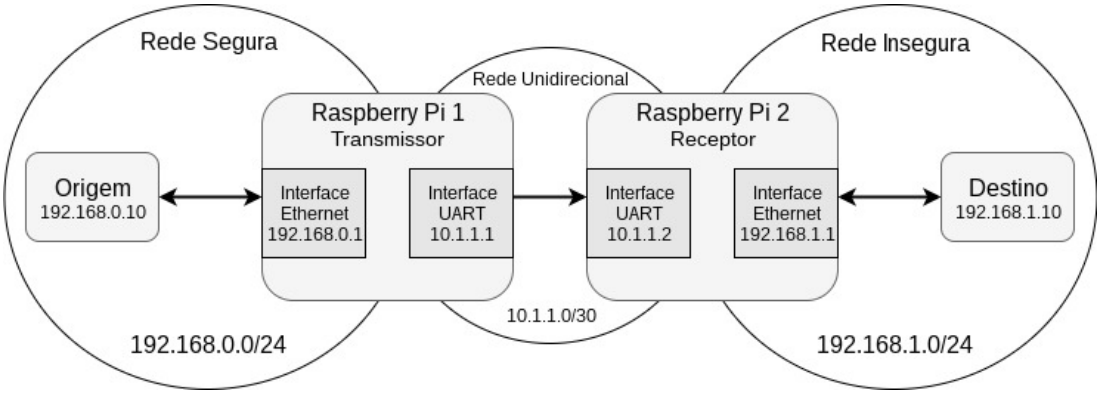

Figura 5.3: Esquema de Rede

possuem nenhum ponto de integração, nem rotas que as interliguem. O único ponto de integração entre as redes é o **Gateway** implementado pelo uso dos dois Raspberry, interligados via interface UART fisicamente unidirecional.

Comunicação com TCP e UDP

Na implementação do *Data Diode* proposto é considerado um cenário onde: (i) existe um único transmissor presente na rede crítica que possui dados a serem transmitidos via conexão TCP; (ii) existe um único destinatário presente na rede insegura; (iii) o único meio de comunicação é um *gateway* unidirecional; (iv) a comunicação deve ser corretamente trafegada via protocolo UDP entre os dispositivos que compõem o Diodo e posteriormente novamente convertida para protocolo TCP na sua entrega ao destinatário final. Desta forma, tal processo deve empregar uma solução de software capaz de organizar, sequenciar e bufferizar todo o fluxo de dados, minimizando o risco de perdas de dados e inconsistência nas informações transmitidas. No modelo funcional apresentado essa é a função do **Intermediário de Conexão**.

Para alcançar esses requisitos de transmissão de dados foram definidas quatro (04) funções:

- A primeira é receber a comunicação TCP que parte da origem na rede crítica.
- A segunda é responsável pela transmissão UDP unidirecional até segundo dispositivo.
- A terceira recebe a comunicação unidirecional transmitida pelo primeiro dispositivo em UDP.

- A quarta e última converte de UDP para TCP e transmite ao destinatário final da comunicação.

Em termos de implementação, cada um dos dois dispositivos Raspberry Pi, responsáveis por desempenhar a função de *Data Diode*, recebeu dois scripts: um script com funções de servidor e outro trabalhando como um cliente de comunicação. É importante esclarecer que na rede crítica considera-se uma única origem de dados - o cliente de um servidor em uma aplicação previamente conhecida - utilizando o protocolo TCP em sua camada de transporte e tendo como destino uma porta pré-estabelecida. Na rede insegura assumiu-se a existência de um único servidor da mesma aplicação requisitada pelo cliente, funcionando sobre o protocolo TCP com uma porta aberta, sendo o destino da comunicação do cliente presente na rede crítica.

O primeiro Raspberry Pi, conectado à rede crítica, possui um script denominado servidorTCP.py (disponível no Anexo B). Este script abre uma porta de comunicação TCP (informada como parâmetro) no mesmo número de porta do destinatário final da comunicação. Com essa porta aberta, o script funciona como um proxy TCP, recebendo os dados transmitidos pelo cliente e armazenando-os em disco sequencialmente. Todo o dado presente na camada de aplicação da pilha TCP/IP presente no pacote de dados é armazenada em um arquivo para posterior transmissão. Vale à pena ressaltar que o armazenamento foi utilizado com o objetivo de garantir a integridade dos dados a serem enviados, considerando principalmente a diferença entre a velocidade de transmissão de dados entre a interface ethernet e a interface UART, evitando assim um estouro de *buffer* ou descarte de dados a serem transmitidos. Esse armazenamento é realizado em um diretório presente no próprio cartão de memória do dispositivo que funciona como o **Armazenamento** descrito no Modelo Funcional.

Ainda no primeiro Raspberry Pi também é executado o script denominado clienteUDP.py (disponível no Anexo B). Esse script é responsável por ler o conteúdo armazenado pelo script servidorTCP.py e enviar esse conteúdo, de forma sequencial, ao segundo Raspberry Pi, sobre o protocolo UDP, utilizando a mesma porta de destino da transmissão de dados inicial. Ao fazer a leitura do arquivo armazenado, verifica-se o tamanho em linhas deste e faz-se uma cópia desse número exato de pacotes armazenados para um segundo arquivo e os dados copiados são apagados do arquivo original. A informação presente no segundo arquivo é transmitida sequencialmente ao segundo Raspberry e após a transmissão de todo arquivo, as informações são novamente buscadas no arquivo original, para gerar um novo conjunto de dados a serem transmitidos. Considerando que a comunicação entre os dispositivos do diodo é fisicamente ponto a ponto, com uma distância mínima entre os equipamentos e com um meio de transmissão confiável, não se faz necessário realizar o sequenciamento dos pacotes através da alteração dos seus

dados, informando números de sequência, no momento do envio.

No segundo Raspberry Pi, conectado à rede insegura, existe um script denominado servidorUDP.py (disponível no Anexo B). Esse script recebe os dados enviados pelo primeiro Raspberry Pi via interface serial UART, em uma comunicação unidirecional, que funciona sobre o protocolo UDP. Enquanto recebe os dados transmitidos pelo seu cliente, armazena-os sequencialmente em um arquivo para posterior transmissão. Nesse caso, o armazenamento não se deve à diferença de velocidade entre as interfaces de rede, mas sim pela necessidades especiais de retransmissão de dados ou a simples análise de dados transmitidos. Esse armazenamento se torna importante, tendo em vista a necessidade de implementar garantias da integridade dos dados transmitidos, e também é realizado no próprio cartão de memória do dispositivo, que representa a unidade **Armazenamento** do Modelo Funcional.

Por fim, no segundo Raspberry Pi, também executa o script clienteTCP.py (disponível no Anexo B). Os dados coletados pelo script servidorUDP.py e armazenados em disco são inicialmente contados, transferidos para um novo arquivo de transmissão e transmitidos via comunicação bidirecional TCP para o destino final da comunicação encontrado na rede insegura. Após a transmissão do arquivo gerado pelo script, o mesmo é criptografado, com o objetivo de impedir que os dados já transmitidos possam ser lidos por entidade não autorizada.

5.2 Requisitos de Segurança

Esta seção descreve os requisitos e as implementações de segurança implementadas para garantir a confidencialidade dos dados e a confiabilidade dos dispositivos envolvidos através das funcionalidades providas pelo TPM e dos recursos providos pelo protocolo SSH. Com base nessas implementações o protótipo demonstrará sua eficácia frente às ameaças possíveis ao ambiente.

5.2.1 Provisionamento

O provisionamento consiste um uma fase pré atestação, onde os valores necessários para comprovar a integridade e autenticidade dos dispositivos são trocados para que os mesmos possam ser posteriormente validados sempre que necessário. Esse processo consiste na geração de um identificador único, uma chave de atestação e um hash baseado nos valores do PCR do chip TPM em uma inicialização considerada segura.

UUID

Inicialmente os dispositivos passam pelo processo de provisionamento visando estabelecer valores de uma inicialização segura para que posteriormente possam ter seus estados testados. Para isso, cada dispositivo deve gerar um UUID (*Universal Unique Identifier*), o identificador único do dispositivo para o sistema remoto que fará a sua atestação, através do comando *tpm_mkuuid* (Figura 5.4).

```
root@servidor-inseguro:~/tpm# tpm_mkuuid server.uuid
root@servidor-inseguro:~/tpm# ls -l
total 4
-rw-r--r-- 1 root root 16 Out 29 19:47 server.uuid
```

Figura 5.4: Execução do comando *tpm_mkuuid*

AIK

Para o UUID do dispositivo também é gerado uma AIK (*Attestation Identity Key*), chave de atestação de identidade, através do comando *tpm_mkaik* (Figura 5.5). Essa chave será utilizada posteriormente, para assinar os dados de atestação gerados no dispositivo, garantindo assim ao dispositivo remoto sua autenticidade. Tanto o UUID quanto a AIK são armazenados no TPM, com o comando *tpm_loadkey* (Figura 5.6). Esses valores serão utilizados tanto na geração do *hash* que identifica o estado dos valores contidos no PCR quanto na atestação remota.

```
root@servidor-inseguro:~/tpm# tpm_mkaik server.bloob server.uuid -z
root@servidor-inseguro:~/tpm# ls -l
total 8
-rw-r--r-- 1 root root 559 Out 29 19:48 server.bloob
-rw-r--r-- 1 root root 304 Out 29 19:48 server.uuid
```

Figura 5.5: Execução do comando *tpm_mkaik*

```
root@servidor-inseguro:~/tpm# tpm_loadkey server.bloob server.uuid
```

Figura 5.6: Execução do comando *tpm_loadkey*

Hash PCR

Com as chaves importadas, um *hash* dos valores contidos no PCR será gerado, pois havendo uma possível mudança nos valores da PCR, este indicará que o processo de inicialização foi modificado, estando diferente da última medição segura.

Através do comando *tpm_getpcrhash* (Figura 5.7) juntamente com os números de identificação dos PCR's utilizados. Os PCR's utilizados nesse processos, também serão os que devem ser avaliados no processo de medição de boot e atestação remota.

```
root@servidor-inseguro:~/tpm# tpm_getpcrhash server.uuid server.hash server.pcrvals
 1 2 3 4 5 6 7 8 9 10
root@servidor-inseguro:~/tpm# ls -l
total 16
-rw-r--r-- 1 root root 559 Out 29 19:48 server.bloob
-rw-r--r-- 1 root root  52 Out 29 19:52 server.hash
-rw-r--r-- 1 root root 431 Out 29 19:52 server.pcrvals
-rw-r--r-- 1 root root 304 Out_29 19:48 server.uuid
```

Figura 5.7: Execução do comando *tpm_getpcrhash*

Ao fim desse processo, os arquivos contendo o UUID do dispositivo e o hash dos PCR's devem ser enviados ao dispositivo remoto que fará a atestação. Sendo trocados, nesse caso, entre o Diodo e os dispositivos de origem e destino. Esses dois arquivos devem ser armazenados, pois serão utilizados posteriormente em na atestação remota.

5.2.2 Atestação Remota

Após a inicialização dos sistemas e o processo de medição de *boot* realizado automaticamente pelo TPM, os dispositivos já estão prontos para iniciar o processo de atestação remota. Como o processo de atestação remota exige a troca de arquivos entre os dispositivos, é sugerido o uso do SSH. A autenticação SSH se dará à partir de uma troca inicial de chaves, para que os scripts envolvidos não precisem carregar senhas de acesso. Toda troca de arquivos via SSH e a execução do script de atestação será realizada por um usuário comum no sistema, ou seja, sem privilégios administrativos dentro do dispositivo local ou remoto. Cada dispositivo deverá gerar um arquivo, denominado *nonce*, em cada processo de atestação, tendo o tamanho exato de 20 bytes, contendo uma sequência numérica aleatória. Este arquivo aleatório, garante que uma nova medição tenha sido gerada no dispositivo remoto, impedindo que uma medição anterior seja utilizada nesse momento, como em um Ataque de Replay. Esse arquivo será transportado ao sistema remoto via SFTP, provido pelo protocolo SSH, e servirá para a geração do arquivo que contém os dados do estado atual dos valores PCR medidos.

Com o arquivo de *nonce*, gerado pelo dispositivo remoto, armazenado localmente, o TPM pode gerar um arquivo de medição (*quote*) através do comando *tpm_getquote* (Figura 5.8). O quote é gerado com base nos valores PCR atuais e no UUID do dispositivo verificado, combinado com o arquivo de dados aleatórios (*nonce*) gerado no dispositivo remoto, assinado digitalmente pela chave AIK que garante sua autenticidade e confiabilidade. O arquivo gerado nesse processo deve

ser enviado para o dispositivo remoto para que haja a atestação remota. Os dois dispositivos devem realizar esse processo com o objetivo da realização de uma atestação mútua.

```
root@servidor-inseguro:~/tpm# tpm_getquote server.uuid /tmp/nonce quote 1 2 3 4 5 6
 7 8 9 10
```

Figura 5.8: Execução do comando *tpm_getquote*

De posse do valor de medição do dispositivo remoto, é executado o comando *tpm_verifyquote* (Figura 5.9) utilizando o UUID do dispositivo remoto, seu *hash* dos valores PCR gerados na fase de provisionamento, o *nonce* gerado localmente no início do processo de atestação e o novo valor enviado pelo dispositivo. Caso os valores estejam íntegros, o comando não reportará nenhum erro ou saída, o que indicará que a atestação remota ocorreu com sucesso. Após o sucesso no processo de atestação, todos os arquivos criados durante essa transação (*nonce* local, *nonce* remoto, *quote* local e *quote* remoto) são apagados do dispositivo, tendo em vista que não serão mais utilizados.

```
root@servidor-inseguro:~/tpm# tpm_verifyquote server.uuid server.hash nonce quote
```

Figura 5.9: Execução do comando *tpm_verifyquote*

Esse processo de atestação ocorre nos dois lados simultaneamente através do *shell script attestation.sh* (disponível no Anexo B). Após o processo de atestação remota, os scripts de servidor e cliente de comunicação são inicializados no Raspberry localizado na rede segura, ativando assim o primeiro lado do Diodo.

VPN

Na rede insegura, o *Data Diode* estabelece uma VPN (*Virtual Private Network*) com o dispositivo que receberá os dados da rede crítica, com o objetivo de garantir a confidencialidade dos dados recebidos. Essa VPN é baseada no protocolo SSH e estabelecida à partir de uma chave de criptografia já trocada entre os dispositivos. Sem esse recurso de criptografia, os dados poderiam ser facilmente interceptados através de um ataque *Man in the Middle* realizado por algum dos computadores na rede insegura. Somente a partir do estabelecimento da VPN é que o *Data Diode* começa a receber os dados e enviá-los ao cliente.

Criptografia dos Dados

Após o envio dos dados ao destinatário, os dois dispositivos Raspberry realizam a criptografia dos dados enviados utilizando o TPM, deixando-os cifrados em um diretório de armazenamento. Todo conjunto de dados transmitido ao próximo

dispositivo é imediatamente criptografado para evitar uma posterior leitura dessas informações transportadas. Para evitar que o armazenamento do dispositivo seja completamente cheio, os dados criptografados há mais de cinco minutos são removidos do dispositivo.

5.3 Testes e Resultados

O protótipo foi submetido a testes em ambiente simulado, com o objetivo de comprovar seu funcionamento e garantir que as ameaças descritas não afetam o cenário em questão. Para os testes assumiu-se que a rede crítica (sem comunicação com a internet) é segura, sendo o principal alvo de testes os dispositivos conectados diretamente na rede insegura. Em todos os testes de transmissão está sendo aplicado um cenário utilizando como base uma comunicação *syslog*[1], funcionando via protocolo TCP, em uma origem na rede segura e chegando a um destino na rede insegura.

Será verificado o Throughput da comunicação unidirecional considerando a velocidade em Mbps e a taxa de perda de dados, tendo por objetivo minimizar as perdas sem denegrir o desempenho da solução. Além disso, a solução será submetida aos ataques descritos no Modelo de Ameaças para atestar seu comportamento frente às ameaças encontradas em um cenário de implementação.

5.3.1 Throughput

Para testar o *throughput* da solução foi utilizado um arquivo com uma carga real de dados a serem transmitidos. Inicialmente, um conjunto de dados *syslog* de 10 MB (mais precisamente 10.398.207 bytes), contendo dados gerados em transmissões anteriores, foi montado para medir a taxa de transferência e a perda de dados no segmento mais crítico da rede entre os dois diodos. Para tal, os dados foram inseridos no *Data Diode* da rede segura no arquivo /dd/514/principal - lido para a transmissão de dados pelo script clienteUDP.py - para que na inicialização do script os dados começassem a ser transmitidos ao segundo *Data Diode*, que os receberá pelo script servidorUDP.py e armazenará os dados no arquivo com o mesmo nome e na mesma localização do primeiro *Data Diode*. Em todos os testes o arquivo era recarregado com o mesmo volume de dados para transmissão e os dados eram apagados no destino. Após os testes preliminares, outros arquivos com tamanhos diferentes também foram testados.

Considerando a discussão na Seção 2.5, foi realizada a alteração do tamanho máximo do datagrama UDP no *Data Diode* através da alteração do MTU (*Maximum Transfer Unit*) da placa de rede *ppp0*, responsável pela transmissão dos

[1] É um protocolo de comunicação de log's via rede TCP/IP

dados UDP. O tamanho máximo de datagrama utilizado foi 210 bytes. O valor de *buffer* não precisou ser alterado, tendo em vista que o sistema operacional utilizado possui um *buffer* para as comunicações UDP maior que 150 * 210 bytes.

Primeira Medição

O primeiro teste executado apresentou uma velocidade de tráfego satisfatória para uma conexão UART, apresentando uma média de 658 Kbps transmitidos, com picos de até 682 Kbps, conforme figura 5.10. Toda medição de tráfego foi obtida pelo software nload.

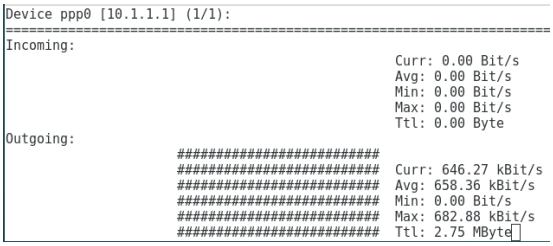

Figura 5.10: Medição de tráfego na interface ppp0

Porém ao analisar o dado recebido no diodo da rede insegura, apenas 1,3 MB (1.356.264 bytes) foram armazenados resultando em uma perda de 9.041.943 bytes, ou seja 86,95% de informação perdida. Um dado importante para análise consiste na transmissão de dados ter demorado apenas 14 segundos para ser concluída. Considerando a quantidade de dados enviados (10.398.207 bytes * 8 = 83.185.656 bits) em 14 segundos, a taxa de transmissão deveria ser no mínimo cerca de 6Mbps, fisicamente impossível para a interface.

Essa grande perda de pacotes se deve à velocidade em que o script envia os dados para a interface de envio, fazendo com que nem toda informação consiga realmente ser transmitida. O tempo aproximado dessa transmissão, levando-se em consideração a taxa de transferência média seria de no mínimo 127 segundos (83.185.656 bits / 658 kbps = 126,42 segundos).

Segunda Medição

Para aumentar o tempo de transmissão e consequentemente diminuir a perda de dados, uma das formas em programação python é adicionar um tempo de espera ao fim de cada envio de dados. Como o script utilizado envia linha a linha dos dados coletados, a cada linha deve haver um tempo de retardo, a fim de garantir que aquela parte da informação seja totalmente transmitida antes da próxima leitura e tentativa.

Para chegar a esse valor, se faz necessário encontrar o tamanho médio dos dados enviados em cada parte da transmissão. Sabendo que o envio acontece linha a linha foi dividido a quantidade total dos dados (10.398.207 bytes) pela quantidade total de linhas do arquivo a ser transmitido, medido pelo comando wc -l, o arquivo contém 79.616 linhas. Portanto cada linha possui em média 130,61 bytes de informação, ou 1044,88 bits. Considerando a velocidade média de transmissão de 658 kbps, chega-se ao valor médio de aproximadamente 630 linhas transmitidas por segundo (658 kbps / 1044,88 bits). Dividindo o tempo de um segundo pela quantidade de linhas transmitidas, de acordo com a velocidade média de transmissão (1s / 630) calcula-se um tempo médio de espera entre as transmissões de cada linha em aproximadamente 0,0016 segundo ou 1,6 milissegundo.

Em cada transmissão foi adicionado um tempo de espera de 1,6 milissegundo através do comando sleep(0,0016) no script clienteUDP.py do diodo transmissor. Após essa alteração foi realizada uma nova medição da transferência do mesmo conjunto de dados, resultando em uma taxa média de transmissão em aproximadamente 530 kbps e pico de 552 kbps.

Nessa segunda medição, o segundo diodo armazenou 10.376.146 bytes de informação, com uma perda de apenas 22.061 bytes, ou seja 0,21% de dados perdidos. O tempo de espera por transmissão se mostrou muito efetivo para o cenário apresentado.

Terceira Medição

Para uma terceira medição, o tempo da função sleep foi dobrado para 3,2 milissegundos com o objetivo de encontrar alguma melhora na integridade dos dados. Uma nova medição da transferência do mesmo conjunto de dados, resultou em uma taxa média de transmissão em aproximadamente 322 kbps e pico de 351 kbps.

O aumento no tempo de espera em milissegundos não apresentou melhora nos resultados da transmissão, sendo a perda de dados de 22.693 bytes, ou seja aproximadamente 0,22%.

Demais Medições

Todas as demais medições utilizaram o tempo de espera do script da Segunda Medição. Foram realizadas três medições com arquivos de aproximadamente 3MB, 5MB, 10MB e 20MB. Os resultados são apresentados na tabela 5.1.

À partir dos dados obtidos percebemos que a perda de informação, se tratando de uma comunicação UDP unidirecional, é mínima, e podem ser consideradas

Tabela 5.1: Medições de Throughput

Medição	Tamanho do Arquivo	Perda	% de Perda
1	3.124.014 bytes	5.963 bytes	0,19%
2	3.124.014 bytes	3.075 bytes	0,10%
3	3.124.014 bytes	4.065 bytes	0,13%
4	5.145.565 bytes	8.263 bytes	0,16%
5	5.145.565 bytes	5.424 bytes	0,11%
6	5.145.565 bytes	7.334 bytes	0,14%
7	10.398.207 bytes	25.124 bytes	0,24%
8	10.398.207 bytes	19.696 bytes	0,19%
9	10.398.207 bytes	24.003 bytes	0,23%
10	20.796.414 bytes	66.105 bytes	0,32%
11	20.796.414 bytes	46.827 bytes	0,23%
12	20.796.414 bytes	50.309 bytes	0,24%

satisfatórias. A comparação desses valores com outros trabalhos relacionados se torna impossível, tendo em vista que os mesmos não apresentam resultados totalmente conclusivos, nem os parâmetros utilizados para os testes.

5.3.2 Negação de Serviço

Um ataque de negação de serviço nesse cenário teria que ser originado de um host dentro da rede insegura ou de um segmento que possua acesso à mesma, considerando que nenhum dos dispositivos envolvidos possuam portas publicadas via NAT para a internet.

Para esse teste foi adicionado à rede um dispositivo com o endereço IP 192.168.1.51, que através do uso do software hping3 iniciou um ataque de força bruta ao diodo (IP 192.168.1.1). O software hping3 teve seu ataque direcionado à porta 22 do diodo, gerando, de acordo com as suas configurações, dez mil pacotes SSH por segundo com o tamanho de 120 bytes, utilizando a flag TCP SYN, conforme a figura 5.11.

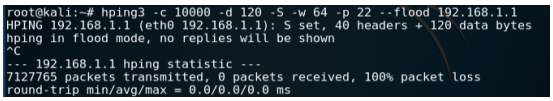

Figura 5.11: Software hping utilizado no teste

Esse ataque se mostrou bastante efetivo, tendo em vista que mesmo bloqueado no firewall do diodo, o atacante conseguiu gerar um grande *overhead* no diodo, impedindo assim a sua comunicação com o servidor remoto. O processo de atestação, que demora menos de 10 segundos em condições normais, ficou

indisponível gerando uma mensagem de conexão SSH perdida nos dois sentidos, conforme as figuras 5.12 e 5.13 .

```
root@dd-receptor:/scripts# su -c /scripts/attestation.sh -s /bin/bash att
estation
Connection closed by 192.168.1.10
lost connection
```

Figura 5.12: Resultado da tentativa de atestação no diodo

```
root@servidor-inseguro:/home/aluno# su -c /scripts/attestation.sh -s /bin/
bash attestation
ssh_exchange_identification: read: Connection reset by peer
lost connection
```

Figura 5.13: Resultado da tentativa de atestação no servidor

Uma efetiva proteção contra esse tipo de ataque consistiria na inteligência do dispositivo de comutação, que interliga os hosts na rede insegura, para identificar o ataque e bloquear a porta do atacante, parando imediatamente o ataque.

5.3.3 Atestação Falsa

Uma atestação falsa presume-se um dispositivo que não realizou o processo de provisionamento no diodo e mesmo assim tenta se atestar como um dispositivo confiável. Vale à pena ressaltar que nesse protótipo todos os acessos via SSH são baseados em chaves rsa e na liberação do endereço IP desse dispositivo no firewall do diodo, portanto, a menos que haja a troca de chaves e a liberação desse endereço no firewall, o dispositivo ficaria impedido de realizar a atestação.

Em um quadro assim, a garantia da não atestação desse dispositivo residiria no fato do diodo não possuir seu UUID nem seu hash PCR para posterior análise. Portanto, mesmo que fosse gerado um nonce para a atestação através de um arquivo de medição (quote), não seria possível realizar a função de verificação, pois não haveriam os arquivos necessários para esse dispositivo.

Nos testes se utilizou um terceiro dispositivo na rede insegura, contendo um chip TPM, *script's* de atestação e simulando ter o mesmo endereço IP do dispositivo real. Conforme log's, percebemos que o processo de atestação falhou impedindo que o mesmo chegasse ao status de dispositivo confiável pelo diodo 5.14.

```
root@dd-receptor:/home/pi# cat /var/log/atestacao.log
2017-10-28 02:35:22 Inicio da Atestacao com o servidor-inseguro
2017-10-28 02:35:31 O dispositivo servidor-inseguro nao foi atestado!
```

Figura 5.14: Log do diodo demonstrando a falha na atestação

Ao submeter o arquivo quote gerado nesse novo dispositivo para sua atestação ao programa tpm_verifyquote, temos como retorno a mensagem de erro apresentada na figura 5.15.

```
root@dd-receptor:/scripts# tpm_verifyquote /tpm/cliente2/cliente2.uuid /t
pm/cliente2/cliente2.hash /home/attestation/nonce /tpm/quote/quote
Error while verifying signature. Error code: TSS_E_FAIL
```

Figura 5.15: Resultado da execução do comando tpm_verifyquote com os dados de um dispositivo não autorizado

Um dispositivo novo somente sera capaz de se atestar, caso seu uuid e seu hash PCR fossem copiados para o dispositivo responsável pela atestação.

5.3.4 Man-in-the-middle

Para os testes dessa ameaça, foi utilizado um terceiro dispositivo na rede insegura denominado aqui atacante. Utilizou-se o software Ettercap para realizar o ataque denominado *ARP Spoofing*, fazendo com que todo o tráfego trocado entre esses dispositivos fosse direcionado ao atacante. Juntamente fora usado o software Tcpdump para realizar a função de *sniffer* de conexão, para receber e armazenar os dados interceptados pelo software Ettercap.

Embora o *ARP Spoofing* tenha obtido sucesso, alterando a tabela ARP dos dispositivos envolvidos (conforme figuras 5.16, 5.17, 5.18 e 5.19), o *sniffing* de pacotes não surtiu o efeito esperado, tendo em vista que toda a comunicação entre o diodo e o servidor é criptografada, impossibilitando a quebra da confidencialidade dos dados, conforme pode-se perceber ao utilizar um software de análise de captura sobre os dados obtidos (figura 5.20).

```
root@dd-receptor:/scripts# arp -an
? (192.168.1.15) at 64:1c:67:69:4b:cf [ether] on eth0
? (192.168.1.51) at 08:00:27:b3:79:0d [ether] on eth0
? (192.168.1.50) at 08:00:27:d9:75:32 [ether] on eth0
? (192.168.1.10) at 08:00:27:cc:1c:dd [ether] on eth0
```

Figura 5.16: Tabela ARP do diodo antes do ataque

```
root@servidor-inseguro:/scripts# arp -an
? (192.168.1.51) em 08:00:27:b3:79:0d [ether] em eth0
? (192.168.1.15) em 64:1c:67:69:4b:cf [ether] em eth0
? (192.168.1.1) em b8:27:eb:d0:c1:23 [ether] em eth0
```

Figura 5.17: Tabela ARP do servidor antes do ataque

5.3.5 Ataque de *Replay*

Considerando que o servidor da rede remota possa ter sido comprometido por um *malware*, que se instalou em seu processo de inicialização, o TPM terá seus valores de PCR comprometidos. A alteração das chaves durante o processo de medição de boot, inviabilizará uma nova atestação, já que o processo de verificação do

```
root@dd-receptor:/scripts# arp -an
? (192.168.1.15) at 64:1c:67:69:4b:cf [ether] on eth0
? (192.168.1.51) at 08:00:27:b3:79:0d [ether] on eth0
? (192.168.1.50) at 08:00:27:d9:75:32 [ether] on eth0
? (192.168.1.10) at 08:00:27:b3:79:0d [ether] on eth0
```

Figura 5.18: Tabela ARP do diodo após do ataque de *man in the middle*

```
root@servidor-inseguro:/scripts# arp -an
? (192.168.1.51) em 08:00:27:b3:79:0d [ether] em eth0
? (192.168.1.15) em 64:1c:67:69:4b:cf [ether] em eth0
? (192.168.1.1) em 08:00:27:b3:79:0d [ether] em eth0
```

Figura 5.19: Tabela ARP do servidor após do ataque de *man in the middle*

quote deste, realizado remotamente no diodo falhará como na Atestação Falsa. Porém através da recuperação de um quote gerado em uma atestação anterior, um atacante poderia tentar uma nova validação deste dispositivo. Porém cada atestação incorre na geração de um novo arquivo aleatório nonce pelo dispositivo que requisita a atestação remota, sendo assim o quote da atestação anterior também foi gerado em conjunto com um nonce recebido durante esse último processo de verificação.

Ao iniciar o processo de atestação e enviar o quote de uma atestação anterior, o diodo reportou em seus log's falha na atestação (figura 5.21). Essa falha se dá pela impossibilidade de utilizar o antigo nonce no dispositivo remoto, tendo em vista que o dispositivo que requer a atestação gera o arquivo aleatoriamente para compor a checagem do quote local. Sendo uma nova atestação, um novo valor aleatório será gerado, impossibilitando que uma atestação local antiga seja novamente utilizada em um ataque de *replay*. A chance de um valor de nonce se repetir em um processo de atestação posterior é de 1 em 10.000.000.000.000.000.000.

5.3.6 Força Bruta

À partir dos dados apresentados quanto às falhas de atestação falsa e ataques de replay, a única tentativa ainda válida para quebra da confidencialidade dos dados seria utilizar um ataque de força bruta. Considerando que toda a comunicação, seja no processo de atestação ou na própria troca dos dados pela rede, está baseada no protocolo SSH, esse protocolo poderia ser o principal alvo de um ataque desse tipo.

Para impedir um ataque de força bruta em senhas de autenticação via protocolo SSH, o cenário foi desenvolvido sobre a premissa de não aceitar autenticação baseada em senhas, sendo a única forma de comunicação a troca de chaves RSA antes mesmo da atestação entre os dispositivos. A única forma para a troca dessas chaves se dá de maneira física nos dispositivos envolvidos, coletando a sua chave pública em um dispositivo de armazenamento móvel e inserindo-a também

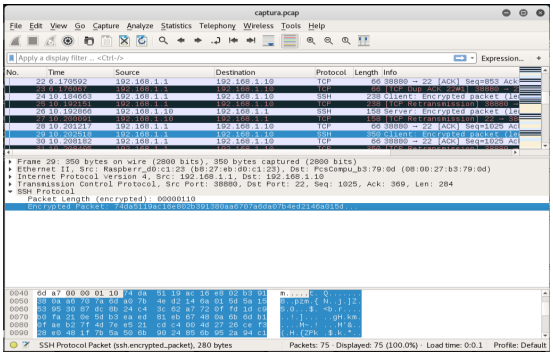

Figura 5.20: Exibição dos dados coletados em um software de análise de pacotes

```
root@dd-receptor:/scripts# cat /var/log/atestacao.log
2017-10-28 03:09:58 Inicio da Atestacao com o servidor-inseguro
2017-10-28 03:10:36 O dispositivo servidor-inseguro nao foi atestado!
```

Figura 5.21: Log escrito após a tentativa de um ataque de *replay*

por acesso físico no dispositivo destinatário da comunicação.

Um ataque de força bruta baseado em senhas, mesmo possuindo a senha correta em sua *password list*, seria completamente inofensivo contra este cenário.

Uma tentativa válida seria um ataque de força bruta sobre as chaves RSA, porém o comprimento de chave utilizado neste protótipo fora o padrão do protocolo, 2048 bit's. Um ataque de força bruta em uma chave com esse comprimento seria completamente ineficiente em um hardware comum, tendo em vista que um ataque de força bruta em chaves RSA utilizaria um algoritmo a ser executado em tempo exponencial e o tamanho das chaves impossibilitaria uma resposta válida em tempo hábil.

Ainda que houvesse a possibilidade de quebra dessas chaves, um algoritmo de troca de chaves poderia ser facilmente implementado, dificultando ainda mais ataques desse tipo em um cenário idêntico.

5.4 Dificuldades Encontradas

A opção da emulação de um chip TPM via software nesse protótipo trouxe várias dificuldades para a execução do cenário. Todos os softwares que implementam as funcionalidades TPM sobre o sistema operacional (como o *TrouSerS* por exemplo) possuem dificuldades na execução de algumas funções com o TPM-Emulator ou outro emulador TPM. Pode-se perceber que o emulador TPM não consegue executar todas as funções de um chip TPM físico. A partir dessa dificuldade os processos de atestação remota tiveram que ser refeitos e executados sem a utilização dos comandos presentes nesses softwares, o que interferiu diretamente no

tempo de execução do trabalho.

O Raspberry Pi não possui uma BIOS (*Basic Input Output System*), o que dificulta a implantação da medição de boot do TPM. Mesmo que houvesse um chip TPM físico conectado ao Raspberry Pi, este não seria capaz de medir a integridade de todo o processo de inicialização do dispositivo. Esta pode ser considerada uma grande falha nesse protótipo, principalmente ao se levar em consideração a possibilidade de um *malware* comprometer o processo de inicialização antes da execução do Sistema Operacional, o que não seria detectado ou alteraria os valores calculados na PCR nesse caso. Portanto uma falha no processo de inicialização resultaria ainda em um boot íntegro e na liberação das chaves contidas no TPM.

Considerando a baixa capacidade de processamento do Raspberry Pi 1 B+ durante a execução do experimento, o equipamento utilizado na rede insegura, que possui a maior quantidade de tarefas executadas (criptografia de arquivos enviados, estabelecimento da VPN além da transmissão dos dados) apresentou um uso constante de 100 % de sua capacidade de processamento. Esse fato inviabilizou uma melhor resposta do protótipo quanto à verificação de *throughput* do cenário, além de amplificar o dano do ataque de negação de serviço.

Capítulo 6

Considerações Finais

De forma a contribuir para o desenvolvimento de soluções contra ameaças à redes industriais críticas, este trabalho apresentou o primeiro modelo funcional para o projeto e desenvolvimento de *Data Diodes* seguros. Um modelo funcional foi apresentado e um protótipo gerado como prova de conceito, deixando claro sua capacidade de atender os requisitos de confidencialidade e integridade. Além disso, vale ressaltar que o protótipo apresentado pode ser utilizado como base de desenvolvimento de soluções de comunicação segura em quaisquer redes industriais críticas onde seja necessário enviar dados em uma única direção.

Com base nesse trabalho, novas soluções de transferência de dados em aplicações TCP, atuais ou legadas, podem ser implementadas utilizando um meio de comunicação unidirecional para o tráfego seguro entre origem e destino. Ficou demonstrado que sistemas de comunicação de dados não necessariamente precisam ser reescritos para se adaptar à comunicação unidirecional, mas o gateway de comunicação pode ser adaptado, a fim de proporcionar uma comunicação segura e o isolamento físico da rede mais crítica.

Através do protótipo foi possível provar que as grandes ameaças ao cenário de aplicação proposto não conseguiram ferir a confidencialidade e a integridade dos dados trafegados, sendo efetiva apenas a negação de serviço como base para a quebra da disponibilidade da solução. O protótipo conseguiu implementar as funções de atestação de hardware disponíveis em um chip TPM e a segurança dos dados trafegados através de criptografia.

Esse trabalho não apresentou tantas comparações em seus resultados obtidos, por não haver uma preocupação com a segurança da rede insegura nos trabalhos até aqui publicados. Os resultados descritos aqui servirão como base de comparação e melhora dos resultados obtidos, como por exemplo a defesa contra ataques de negação de serviço e a solução contra os problemas de *overhead* causados ao dispositivo Raspberry Pi B + por conta de suas funções de criptografia.

6.1 Trabalhos Futuros

A partir do trabalho desenvolvido aqui, novos cenários de comunicação seguros podem ser desenvolvidos. Considerando que a prototipação interligou apenas um dispositivo na rede crítica com um dispositivo de destino em uma rede insegura, será possível imaginar um cenário onde várias origens possam se comunicar com variados destinos dentro de uma única rede insegura ou até mesmo na internet. O diodo nesse caso poderá funcionar como um dispositivo de NAT, gerenciando as conexões à partir das portas de comunicação, estabelecendo transações simultâneas entre múltiplos dispositivos, sempre em sentido unidirecional, mantendo os requisitos de segurança e criptografia. Toda a atestação mútua pode ser concentrada entre as múltiplas origens e destinos no diodo. Uma cadeia de certificação digital pode ser implementada a partir do TPM contido no diodo, gerando e controlando certificados digitais individuais para seus destinos na rede insegura.

Levando em consideração a segurança dos dados trafegados entre os dois diodos presentes no cenário, pode-se implementar um esquema de criptografia simétrica e assimétrica com o objetivo de garantir ao diodo da rede insegura que o dado fora realmente enviado pelo diodo da rede segura por meio de uma assinatura digital e garantir ao diodo da rede segura que seus dados serão lidos apenas pelo diodo da rede insegura através da criptografia dos dados enviados. Esse cenário de implementação causaria um grande *overhead* nos dispositivos, sendo impossível a sua implementação através dos equipamentos utilizados nesse trabalho. Um projeto de um hardware capaz de suportar essas funções, utilizando tecnologias disponíveis no mercado também se tornaria fruto de pesquisa nesse trabalho futuro.

Um trabalho futuro importante também será a implementação de uma prova de conceito da arquitetura Dispositivo único com chip FPGA. Essa arquitetura possui um grande potencial de resolver as questões de throughput e perda de dados na comunicação unidirecional, porém é um cenário completamente diferente das implementações atuais de *Data Diodes* e portanto deve receber uma atenção especial na definição de quais dispositivos podem ser utilizados. Além disso, se torna imprescindível para esse método um aprofundamento na configuração de dispositivos FPGA e na escolha de um chip capaz de prover um real isolamento físico entre as interfaces de rede e operações de leitura e escrita em dispositivos de armazenamento.

Um método de recuperação dos dados armazenados, que possam ter passado por problemas no envio deve ser levado em consideração em uma nova implementação, visando uma nova garantia sobre a integridade da informação transferida. Além do armazenamento seguro (criptografado) das informações na fila de transmissão, durante seu processo de gravação e a descriptografia apenas na transmissão, se aproveitando dos recursos do TPM.

Um console de configuração remota seguro, acessível apenas à partir da rede segura, para habilitar novas portas de comunicação e autorizar novos dispositivos à atestação. Onde a operação do par de diodos possa ser completamente modelada de acordo com o cenário de implementação por um administrador de rede.

Por meio da avaliação das técnicas de aprendizagem de máquina disponíveis, uma pesquisa profunda sobre métodos aplicáveis poderia servir como base para um diodo que se adapte a estruturas de redes existentes, ou simplesmente trabalhando como *gateway* personalizado de um servidor, gerenciando a comunicação de toda a rede com ele e estabelecendo o tráfego unidirecional, protegendo seus dados. Um dispositivo aplicável em qualquer ambiente de rede ou servidor possui potencial para se tornar a solução definitiva para proteção de backup's e log's centralizados, principalmente em um momento onde *ransonware* se apresenta como uma ameaça real e recorrente, encripitando servidores inteiros em *datacenter's* pelo mundo.

Referências Bibliográficas

[1] S. Huang, C. J. Zhou, S. H. Yang, and Y. Q. Qin, "Cyber-physical system security for networked industrial processes," *International Journal of Automation and Computing*, vol. 12, no. 6, pp. 567–578, 2015.

[2] N. Moreira, E. Molina, J. Lázaro, E. Jacob, and A. Astarloa, "Cyber-security in substation automation systems," *Renewable and Sustainable Energy Reviews*, vol. 54, pp. 1552–1562, 2016.

[3] R. T. Barker and C. J. Cheese, "The application of data diodes for securely connecting nuclear power plant safety systems to the corporate IT network," *System Safety, incorporating the Cyber Security Conference 2012, 7th IET International Conference on*, pp. 1–6, 2012.

[4] B.-s. Jeon and J.-c. Na, "A Study of Cyber Security Policy in Industrial Control System using Data Diodes," pp. 314–317, 2016.

[5] W. A. Conklin, "State Based Network Isolation for Critical Infrastructure Systems Security," *Proceedings of the 48th Hawaii International Conference on System Sciences (HICSS-48)*, pp. 2280–2287, 2015.

[6] J. Melorose, R. Perroy, and S. Careas, "Unidirectional Secure Information Transfer via RabbitMQ," *Statewide Agricultural Land Use Baseline 2015*, vol. 1, no. December, 2015.

[7] H. Lin, "Research on Packet Loss Issues in Unidirectional Transmission," *Journal of Computers*, vol. 8, no. 10, pp. 2664–2671, 2013.

[8] Y. Heo, B. Kim, D. Kang, and J. Na, "A Design of Unidirectional Security Gateway for Enforcement Reliability and Security of Transmission Data in Industrial Control Systems," pp. 310–313, 2016.

[9] H. Okhravi and F. T. Sheldon, "Data diodes in support of trustworthy cyber infrastructure," *Proceedings of the Sixth Annual Workshop on Cyber Security and Information Intelligence Research - CSIIRW '10*, p. 1, 2010.

[10] Y.-c. Oh, M.-r. Han, Y. Shin, and J.-b. Kim, "A Study on the Communication Agent Model for One-way Data Transfer System," vol. 9, no. 10, pp. 161–168, 2015.

[11] V. Arkhangelskii, A. Epishkina, V. Kalmytov, and K. Kogos, "Secure One-Way Data Transfer," pp. 392–395, 2016.

[12] J. Winter and K. Dietrich, "A hijacker's guide to communication interfaces of the trusted platform module," *Computers and Mathematics with Applications*, vol. 65, no. 5, pp. 748–761, 2013.

[13] J. D. Osborn and D. C. Challener, "Trusted platform Module evolution," *Johns Hopkins APL Technical Digest (Applied Physics Laboratory)*, vol. 32, no. 2, pp. 536–543, 2013.

[14] S. Thombre, "Modelling of UDP throughput," *IEEE Region 10 Annual International Conference, Proceedings/TENCON*, pp. 482–487, 2017.

[15] S. H. Ali, S. A. Nasir, and S. Qazi, "Impact of router buffer size on TCP/UDP performance," *2013 3rd IEEE International Conference on Computer, Control and Communication, IC4 2013*, no. January 2015, 2013.

[16] Y. Heo and J. Na, "Development of unidirectional security gateway appliance using intel 82580EB NIC interface," *2016 International Conference on Information and Communication Technology Convergence, ICTC 2016*, pp. 1194–1196, 2016.

[17] E. Nilsson and M. Sundberg, "Emulation of tpm on raspberry pi," 2015. Student Paper.

[18] J. BIOLCHINI, "Systematic Review in Software Engineering.," tech. rep., COPPE/UFRJ, 2005.

[19] S. N. MAFRA and G. H. TRAVASSOS, "Estudos Primários e Secundários Apoiando a Busca por Evidência em Engenharia de Software.," tech. rep., PESC - COPPP/UFRJ, 2006.

[20] B. KITCHENHAM, "Procedures for Performing Systematic Reviews.," tech. rep., Software Engineering Group, Department of Computer Science, Keele University, 2004.

[21] V. R. BASILI, G. CALDIERA, and H. D. ROMBACH, "The Experience Factory.," 1994.

[22] H. J. Lee and D. Won, "Protection profile for unidirectional security gateway between networks," *International Journal of Security and its Applications*, vol. 7, no. 6, pp. 373–384, 2013.

[23] J. Wang, J. Liu, S. Yang, and M. Zhang, "Integrated trusted protection technologies for industrial control systems," *2016 18th International Conference on Advanced Communication Technology (ICACT)*, pp. 70–75, 2016.

[24] B. Bertholon, S. Varrette, and P. Bouvry, "CERTICLOUD: A novel TPM-based approach to ensure cloud IaaS security," *Proceedings - 2011 IEEE 4th International Conference on Cloud Computing, CLOUD 2011*, pp. 121–130, 2011.

[25] S.-C. Hsueh and C.-C. Kuo, "SECURING MOBILE ACCESS OF CONFIDENTIAL DOCUMENTS BY INTEGRATING TRUSTED COMPUTING PLATFORMS WITH DIGITAL RIGHTS MANAGEMENTS," *Statewide Agricultural Land Use Baseline 2015*, vol. 1, pp. 2–5, 2015.

[26] H. Zhang, Z. Qin, and Q. Yang, "Design and implementation of the TPM chip J3210," *Proceedings - 3rd Asia-Pacific Trusted Infrastructure Technologies Conference, APTC 2008*, pp. 72–78, 2008.

[27] C. Kim, "Cyber-Resilient Industrial Control System with Diversified Architecture and Bus Monitoring," pp. 11–16, 2016.

Apêndice A

Revisão Sistemática da Literatura

Este Apêndice tem como objetivo descrever a Revisão Sistemática sobre os esquemas de segurança da informação aplicados em Redes Industriais e a utilização de *Data Diodes* e TPM. O protocolo utilizado segue a especificação proposta por: Biolchini et al.[18] , Mafra e Travassos [19], e Kitchenham [20].

A.1 Objetivo

O objetivo deste estudo será esquematizado a partir do paradigma GQM (goal, question, and metric) descrito em Basili et al.[21]:

Tabela A.1: Tabela GQM

Analisar	Esquemas de segurança da informação aplicados em Redes Industriais
Com o propósito de	Identificar
Com relação a	Utilização de Data Diodes e TPM
Do ponto de vista do	Pesquisador
No contexto	Pesquisas acadêmicas e científicas em segurança da informação

A.2 Questão de Pesquisa

Como são implementados Data Diodes e TPM em redes de ambientes industriais?

A.3 Fonte de Pesquisa

Para as fontes de pesquisa foram adotados os critérios a seguir:

- Consulta de artigos em bibliotecas digitais, congressos, simpósios, revistas;
- Disponibilidade de consulta de artigos através da web;
- Preferencialmente presença de mecanismos de busca através de palavras-chaves;
- Ter os estudos disponíveis em inglês ou português;

A fonte de pesquisa foi acessada via web, através de máquinas de buscas e a inserção manual também foi utilizada. A fonte utilizada foi o Scopus (http://www.scopus.com).

A.4 Critérios de Inclusão e Exclusão de Artigos

Os seguintes critérios de inclusão e exclusão foram utilizados na revisão:

- (I) Estudo aplica um esquema de segurança da informação para redes industriais;
- (I) Apresentar a utilização de um data diode ou de um hardware TPM;
- (E) Estudo não estar disponível;
- (E) Não estar escrito em inglês ou português;
- (E) Não atender critérios de inclusão;
- (E) Trabalho duplicado;

Antes da primeira análise deverão ser eliminados os itens em duplicidade. Na primeira análise (primeiro filtro) serão considerados os seguintes itens: Título, Resumo (Abstract) e Palavras-Chave.

Em uma segunda análise serão extraídos os seguintes dados: Título, Autores, Ano, Conferência, Abstract e Texto.

A.5 Strings de Busca

Para a filtragem dos resultados, as seguintes string's de busca foram testadas, afim de encontrar um número de resultados que fosse relevante e possibilitasse a revisão:

Tabela A.2: Strings

ID	String	Retorno
1	(("security information"OR "data security"OR "security communitation") AND (network OR "data transfer") AND industrial) OR ("data diode"OR "unidirectional gateway"OR "unidirectional data transfer"OR "One way data transfer"OR "unidirectional security gateway") OR (TPM AND ("security"AND "information"))	3.690
2	(("data diode"OR "unidirectional gateway"OR "unidirectional data transfer"OR "one way data transfer"OR "unidirectional security gateway") OR (TPM AND ("security"AND "information"))) AND (industrial OR industry OR scada)	323
3	(("data diode"OR "unidirectional gateway"OR "unidirectional data transfer"OR "one way data transfer"OR "unidirectional security gateway"OR "one way gateway") OR (TPM AND ("security"AND "information"AND encryption))) AND (industrial OR industry)	91

A.6 Condução da Revisão

A String com o ID 3 (conforme Tabela A.2) retornou 91 artigos os quais foram importados para a ferramenta StArt, não sendo detectada nenhuma duplicidade.

Em pesquisas anteriores, foram encontrados quatro artigos não indexados na busca automática. Tendo em vista a importância dos textos para o trabalho, foi necessária a inserção manual dos seguintes trabalhos na Revisão Sistemática:

- Secure One-way data transfer [11]
- Unidirectional Secure Information Transfer via RabbitMQ [6]
- A Study on the Communication Agent Model for One-way Data Transfer System [10]
- Protection Profile for Unidirectional Security Gateway between Networks [22]

Após o primeiro filtro foram excluídos 64 artigos e no segundo filtro foram excluídos mais 15. Portanto ao fim do processo de filtragem foram extraídos 16 artigos, sendo:

- A design of unidirectional security gateway for enforcement reliability and security of transmission data in industrial control systems [8]

- A study of cyber security policy in industrial control system using data diodes [4]
- Cyber-security in substation automation systems [2]
- State based network isolation for critical infrastructure systems security [5]
- Research on data leak protection technology based on trusted platform [23]
- Research on packet loss issues in unidirectional transmission [7]
- Trusted platform Module evolution [13]
- A hijacker's guide to communication interfaces of the trusted platform module [12]
- CERTICLOUD: A novel TPM-based approach to ensure cloud IaaS security [24]
- Data diodes in support of trustworthy cyber infrastructure [9]
- Securing mobile access of confidential documents by integrating trusted computing platforms with digital rights managements [25]
- Design and implementation of the TPM chip J3210 [26]
- Secure One-way data transfer [11]
- Unidirectional Secure Information Transfer via RabbitMQ [6]
- A Study on the Communication Agent Model for One-way Data Transfer System [10]
- Protection Profile for Unidirectional Security Gateway between Networks [22]
- Development of unidirectional security gateway appliance using intel 82580EB NIC interface [16]
- Cyber-resilient industrial control system with diversified architecture and bus monitoring [27]

A.7 Resultados Obtidos

Através desta Revisão Sistemática, se torna possível obter o seguintes resultados: Há um grande aumento na quantidade de artigos publicados sobre o problema da Segurança em Redes Industriais a partir de 2010, com alguns esquemas de segurança propostos. A tecnologia mais sugerida por esses artigos é o Data Diode em substituição aos firewall's tradicionais. Porém nenhum desses artigos sugere a utilização de Data Diodes combinados com o hardware criptográfico TPM para a atestação mútua de dispositivos e também falham na passagem de tráfego TCP entre as redes segmentadas por gateways unidirecionais.

A.8 Lições aprendidas

Essa Revisão contribui com as seguintes lições aprendidas:

1. Embora o problema seja discutido de forma abrangente, ainda não foram esgotadas todas as possibilidades de implementações de segurança da informação.

2. O hardware criptográfico TPM está em um estágio maduro de evolução, e as características encontradas nele já contemplam todas as necessidades encontradas no problema proposto.

3. *Data diodes* possuem ainda restrições que precisam ser tratadas. No esquema proposto na maior parte dos artigos encontrados há o problema da não retransmissão de dados, comprometendo assim a integridade das informações transmitidas. Em outros esquemas, a retransmissão é tratada, porém um único tipo de tráfego é aceito, impedindo assim o funcionamento adequado de vários protocolos.

Apêndice B

Códigos

Este Apêndice tem como objetivo disponibilizar todos os códigos utilizados nesse trabalho.

B.1 Script servidorTCP.py

```
import socket
import sys
import os

# Inicia o Socket TCP para receber os dados
try:
	mysock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
except socket.error:
	print "Erro ao criar o socket"
	sys.exit(1)

# Recebe a porta a ser utilizada como parâmetro e configura o IP
porta = int(sys.argv[1])
host = "192.168.0.1"
# Inicia o socket no IP e porta determinados
mysock.bind((host, porta))
mysock.listen(1)

# Cria o diretório que recebera os dados
os.system("mkdir /dd/"+ str(porta))
while 1:
	# Recebe as mensagens
```

```
        con, cliente = mysock.accept()
        while 1:
                msg = con.recv(1024)
                # Abre o arquivo onde os dados serão armazenados (se não existir e
criado aqui)
                arq = open('/dd/' + str(porta) + '/principal','ab')
                # Escreve e fecha o arquivo
                arq.write(msg)
                arq.close()
        # Fecha o Socket
        con.close()
        mysock.close()
```

B.2 Script clienteUDP.py

```
import os
import commands
import socket
import sys
# DIODO 1 - Origem
# Socket UDP lado Rede Segura
# Endereço IP do segundo Diodo
host = '10.1.1.2'
# Porta do segundo Diodo, recebida por parâmetro
porta = int(sys.argv[1])
# Inicia o Socket UDP
mysock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
# Indica o arquivo onde os dados serao lidos para envio ao destino final
arquivo = "/dd/"+ str(porta) + "/principal"
# Cria o diretório para os dados ja enviados
os.system("mkdir /dd/"+ str(porta) + "/enviado")
while True:
        # Verifica o n mero de linhas a serem lidas e enviadas
        linhas = commands.getoutput("cat "+ arquivo +  wc -l")
        numlinhas = int(linhas)
        # Se houverem linhas para envio realiza a transmissão
        if numlinhas > 0:
                # Os dados sao transferidos para um arquivo que ser lido e enviado,
esse arquivo tera como nome a data atual
```

```
            from datetime import datetime
            now = datetime.now()
            arquivo_ envio = "/dd/"+ str(porta) + "/"+ str(now.day) + str(now.hour)
+ str(now.minute) + str(now.second)
            os.system("cat "+ arquivo +  sed -n '1,"+ linhas + "p' > "+ ar-
quivo_envio)
            os.system("sed -i '1,"+ linhas + "d' "+ arquivo)
            rep = 0
            arq = open(arquivo_envio, 'rb')
            # Realiza o envio dos dados linha a linha
            while rep < numlinhas:
                  msg = arq.readline()
                  mysock.sendto(msg,(host,porta))
                  # Adiciona o tempo de espera para o envio do pacote
                  sleep(0,0016)                    rep +=1
            # Fecha o arquivo
            arq.close()
            # Move os dados ja enviados para outro diretório
            os.system("mv "+ arquivo_envio + "/dd/"+ str(porta) + "/envi-
ado/")
```

B.3 Script servidorUDP.py

```
import socket
import sys
import os

# Inicia o Socket UDP para receber os dados
try:
      mysock = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
except socket.error:
      print "Erro ao criar o socket"
      sys.exit(1)

# Recebe a porta a ser utilizada como parâmetro e configura o IP
porta = int(sys.argv[1])
host = "10.1.1.2"
# Inicia o socket no IP e porta determinados
mysock.bind((host,porta))
```

```
# Cria o diretório que receberá os dados
os.system("mkdir /dd/"+ str(porta))
while 1:
	# Recebe as mensagens
	msg = mysock.recv(1024)
	# Abre o arquivo onde os dados serao armazenados (se nao existir e criado
aqui)
	arq = open('/dd/' + str(porta) + '/principal','ab')
	# Escreve e fecha o arquivo
	arq.write(msg)
	arq.close()

# Fecha o Socket
mysock.close()
```

B.4 Script clienteTCP.py

```
import os
import commands
import socket
import sys
# DIODO 2 - Destino
# Socket TCP lado Rede Insegura
# Endereço IP do Servidor de Destino
host = '10.0.2.2'
# Porta do serv de Destino, recebida por parametro
porta = int(sys.argv[1])
# Inicia o Socket TCP
tcp = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
tcp.connect((host, porta))
# Indica o arquivo onde os dados serão lidos para envio ao destino final
arquivo = "/dd/"+ str(porta) + "/principal"
# Cria o diretório para os dados já enviados
os.system("mkdir /dd/"+ str(porta) + "/enviado")
while True:
	# Verifica o número de linhas a serem lidas e enviadas
	linhas = commands.getoutput("cat "+ arquivo +  wc -l")
	numlinhas = int(linhas)
```

```
        # Se houverem linhas para envio realiza a transmissão
        if numlinhas > 0:
                # Os dados são transferidos para um arquivo que será lido e enviado,
esse arquivo tera como nome a data atual
                from datetime import datetime
                now = datetime.now()
                arquivo_envio = "/dd/"+ str(porta) + "/"+ str(now.day) + str(now.hour)
+ str(now.minute) + str(now.second)
                os.system("cat "+ arquivo +  sed -n '1,"+ linhas + "p' > "+ ar-
quivo_ envio)
                os.system("sed -i '1,"+ linhas + "d' "+ arquivo)
                rep = 0
                arq = open(arquivo_ envio, 'rb')
                # Realiza o envio dos dados linha a linha
                while rep < numlinhas:
                        msg = arq.readline()
                        tcp.send(msg)
                        rep +=1
                # Fecha a conexão e o arquivo
                tcp.close
                arq.close()
                # Criptografa o arquivo com os dados já enviados, utilizando o TPM
                os.system("tpm_sealdata -i "+ arquivo_envio + -o "+ arquivo_envio
+ ".cr -z")
                # Remove o arquivo não criptografado
                os.system("rm -r "+ arquivo_ envio)
                # Transfere o arquivo já criptografado para o diretório de dados en-
viados
                os.system("mv "+ arquivo_envio + ".cr /dd/"+ str(porta) + "/en-
viado/")
```

B.5 Script attestation.sh

```
#!/bin/bash

#Escrita de LOG echo $(date +%F%T) "Inicio da Atestacao com o servidor-
inseguro»> /var/log/atestacao.log
# Gerar um nonce
echo $(($RANDOM % 10))$(($RANDOM % 10))$(($RANDOM % 10))$(($RAN-
```

```
DOM % 10))$(($RANDOM % 10))$(($RANDOM % 10))$(($RANDOM % 10))$(($RAN-
DOM % 10))$(($RANDOM % 10))$(($RANDOM % 10))$(($RANDOM % 10))$(($RAN-
DOM % 10))$(($RANDOM % 10))$(($RANDOM % 10))$(($RANDOM % 10))$(($RAN-
DOM % 10))$(($RANDOM % 10))$(($RANDOM % 10))$(($RANDOM % 10))
> /home/attestation/nonce

# Enviar o nonce
scp /home/attestation/nonce attestation@servidor-inseguro:/tpm/nonce

# Aguarda o nonce do host remoto
var=1
while [ $var -ne 0 ]
do
        if [ "$(ls -A /tpm/nonce)"]
        then
                var=0
        else
                sleep 2
        fi
done

# Recebeu o nonce e geração quote
tpm_getquote /tpm/diodo2/diodo2.uuid /tpm/nonce/nonce /home/attestation/quote
1 2 3 4 5 6 7 8 9 10

# Envia o quote para verificação
scp /home/attestation/quote attestation@servidor-inseguro:/tpm/quote

# Verifica se recebeu o quote
var=1
while [ $var -ne 0 ]
do
        if [ "$(ls -A /tpm/quote)"]
                then
                        var=0
                else
                        sleep 2
        fi
done
```

```
# Verifica o quote remoto e atesta o dispositivo
tpm_verifyquote /tpm/cliente2/cliente2.uuid /tpm/cliente2/cliente2.hash /home/attestation/nonce
/tpm/quote/quote
if [ $? -eq 0 ]
      then
            #Escrita de LOG                echo $(date +%F%T) "O dispositivo
servidor-inseguro foi atestado com sucesso!»> /var/log/atestacao.log        else
            #Escrita de LOG                echo $(date +%F%T) "[ERRO] O dispo-
sitivo servidor-inseguro nao foi atestado»> /var/log/atestacao.log                #
limpa o quote e o nonce
            rm -r /tpm/nonce/nonce /tpm/quote/quote /home/attestation/quote
/home/attestation/nonce
            # inicia o processo novamente
            /scripts/attestation.sh
fi
```

B.6 Script inicia VPN.sh

```
#!/bin/bash
var=1
while [ $var -ne 0 ]
do
      # Verifica se o Sistema ja foi atestado
      tpm_verifyquote /tpm/cliente2/cliente2.uuid /tpm/cliente2/cliente2.hash
/home/attestation/nonce /tpm/quote/quote
      if [ $? -eq 0 ]
            then
                  echo "Dispositivo remoto atestado!"
                  var=0
                  # inicia a VPN
                  ip tuntap add tun0 mode tun
                  ip addr add 10.0.2.1/30 dev tun0
                  ip link set dev tun0 up
                  # Espera 20 segundos, visa garantir o fim da atestação no host
remoto
                  sleep 20
                  ssh -f -w0:0 root@servidor-inseguro true
                  # libera o firewall
```

```
                iptables -A INPUT -s 10.0.2.2 -j ACCEPT
                # inicia o cliente e o servidor
                python /scripts/server.py 514 &
                python /scripts/client.py 514 &
                # Limpa o quote e o nonce recebidos e enviados
                rm -r /tpm/nonce/nonce /tpm/quote/quote /home/attestation/quote
/home/attestation/nonce
            else
                echo "Nao atestou"
                # Aguarda 5 segundos e reinicia a verificação
                sleep 5
        fi
done
```