Comparação de cifras do TLS 1.3 em uma comunicação entre sistemas embarcados e um servidor web

TLS 1.3 cipher suite comparison in a communication between embedded systems and a web server

Augusto Cézar Ferreira *, Natássya Barlate Floro da Silva†

RESUMO

Sistemas embarcados são a combinação entre hardware e software que tem como propósito realizar funções específicas repetidamente. Comumente utilizam de redes de computadores para se comunicar com outros dispositivos e, devido a isto, podem exigir requisitos de segurança, como confidencialidade, autenticidade e integridade das informações. Protocolos de segurança são aplicados em dispositivos para garanti-los, porém, para sistemas embarcados, poucas técnicas são efetivamente aplicadas. Este artigo tem como objetivo a avaliação e comparação do tempo de execução do protocolo TLS na versão 1.3 na comunicação entre dois sistemas embarcados diferentes (nodeMCU e Raspberry Pi3) e um servidor web. Foram consideradas três cifras, sendo elas TLS_AES_128_GCM_SHA256, TLS_AES_256_GCM_SHA384 e TLS_CHACHA20_POLY1305_SHA256, sendo possível verificar qual dispositivo e qual cifra possui o menor tempo de comunicação. Os resultados mostraram que, em geral, os tempos de comunicação para o nodeMCU são maiores devido à arquitetura, enquanto a cifra que possui menor tempo é a CHACHA20_POLY1305_SHA256 sendo cerca de 22% e 28% mais rápida que o AES_128_GCM_SHA256 e o AES_256_GCM_SHA384, respectivamente.

Palavras-chave: Segurança. TLS 1.3. Sistemas Embarcados.

ABSTRACT

Embedded systems are the combination between hardware and software designed to perform specific functions. They commonly use computer networks to communicate with other devices, and because of this, they may demand security requirements, such as confidentiality, authenticity and integrity of information. Security protocols are applied to devices to guarantee them. However, for embedded systems, few techniques are effectively applied. This paper aims to evaluate and compare the TLS protocol runtime in version 1.3 for the communication between two different embedded systems (nodeMCU and Raspberry Pi3) and a web server. Three ciphers, TLS_AES_128_GCM_SHA256, TLS_AES_256_GCM_SHA384 and TLS_CHACHA20_POLY1305_SHA256, were evaluated to verify which device and which cipher has the shortest communication time. The results showed that, in general, the communication times for nodeMCU are longer due to the architecture, while the cipher that has the shortest time is CHACHA20_POLY1305_SHA256 being about 22% and 28% faster than AES _128_GCM_SHA256 and AES_256_GCM_SHA384, respectively.

Keywords: Security. TLS 1.3. Embedded Systems.

* ; augustof@alunos.utfpr.edu.br.
† ; natassyasilva@utfpr.edu.br.

1 INTRODUÇÃO

A forma como o ser humano se comunica e interage com o meio vem sendo influenciada pelas diferentes tecnologias que surgem conforme o passar dos anos, tais estas que tem como objetivo melhorar a comunicação, automatizar tarefas de cunho intelectual e/ou mecânicas. Tendo isto em vista, podem ser destacados os sistemas embarcados como a combinação de software que realiza tarefas lógicas previamente programadas e um hardware capaz de prover os recursos necessários como memória, processamento e energia, além de possibilitar acoplamento de módulos capazes de realizar tarefas mecânicas (BARR, 1999). Estes são utilizados devido ao seu baixo custo, fácil mobilidade e manuseio e suporte a diversas interfaces de comunicação como WiFi, Bluetooth, Ethernet e Serial, aumentando ainda mais a gama de possíveis aplicações (SILVA et al., 2013).

Com o cenário atual da tecnologia e o surgimento de grandes massas de dados, há grande preocupação em relação à segurança, como na garantia das propriedades de confidencialidade, autenticidade e integridade dos dados (STALLINGS et al., 2012). Contudo, as soluções de segurança são especificadas em sua maioria para sistemas de propósito geral, não levando em consideração as restrições de sistemas embarcados (HAMEED; ALOMARY, 2019).

Atualmente para que seja possível haver uma transmissão segura com confidencialidade, integridade e autenticidade dos dados, protocolos de segurança são aplicados, sendo atualmente o mais atualizado o TLS (*Transpor Secure Layer*) na versão 1.3 (ARFAOUI et al., 2019). O TLS também se destaca como um protocolo de segurança amplamente empregado para aplicações na Internet e vêm sendo associado também aos protocolos específicos para dispositivos IoT (DATTA; SHARMA, 2017). Logo, é importante mensurar os impactos do protocolo na comunicação para estes dispositivos, principalmente diante de suas limitações. Dessa forma, o problema de pesquisa pode se evidencia pela pergunta: Qual é o impacto do protocolo TLS 1.3 na comunicação de sistemas embarcados?

Este trabalho apresenta uma avaliação do TLS 1.3 em dois sistemas embarcados através do conjunto de ferramentas criptográficas wolfSSL. Os sistemas utilizados foram um de menor capacidade de memória e processamento (ESP8266 nodeMCU) e outro mais robusto (Raspberry Pi 3) para o cenário de comunicação com um servidor web desenvolvido em nodeJs hospedado na plataforma Heroku. O objetivo central é verificar a viabilidade e comparar o impacto do protocolo em relação aos tempos gastos para realizar a comunicação segura para as suas diferentes cifras, visando sempre a disponibilidade, confidencialidade, integridade e autenticidade das mensagens trocadas na comunicação.

2 MÉTODO

2.1 Servidor web

Um servidor web foi desenvolvido em nodeJS, uma linguagem que permite a execução de códigos *javascript* fora do navegador web para guardar os dados enviados pelos sistemas embarcados. Rotas foram definidas para os diferentes tipos de dados coletados, com uma rota para armazenar o tempo de *handshake* e outra para o tempo da comunicação, além de uma rota para a coleta das coordenadas enviadas. Foi utilizado também um banco de dados não relacional para o armazenamento dos dados, o mongoBD, escolhido por sua facilidade de operação e por ter suas inserções e seleções apresentadas no formato JSON *JavaScript Object Notation* (BANKER et al., 2016) que é comumente utilizado em requisições web (HOWS; MEMBREY; PLUGGE, 2019).

Para a hospedagem do servidor, foi utilizado o *Heroku*, que é uma plataforma de hospedagem em nuvem que permite realizar o *deploy* da aplicação, dando a opção de realizá-lo automaticamente ao realizar alterações no repositório do *GitHub* do projeto. A plataforma possibilita configurações de domínio, certificados e visualização de métricas de acesso de maneira fácil e gratuita (HANJURA, 2014).

O *Heroku* disponibiliza a utilização do TLS em seu plano pago, por isto para aplicar o protocolo no servidor, foi utilizado o Cloudflare, que presta serviços de segurança e DNS (*Domain Name System*), localizados entre o visitante e o provedor do usuário, agindo como um *proxy* reverso (DEWI; RUSYDI; IMAM, 2019). No Cloudflare, foi adicionado o alvo do DNS fornecido pelo Heroku, com um certificado autoassinado para a comunicação por meio do TLS 1.3.

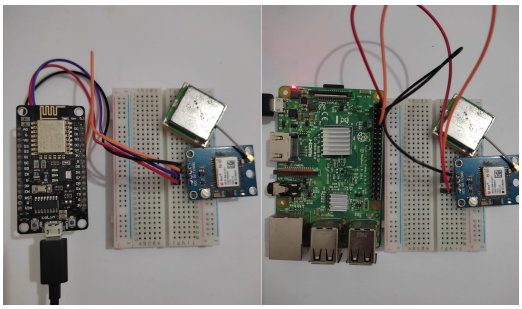
2.2 Sistemas Embarcados

O primeiro dispositivo usado nos experimentos é o nodeMCU, uma placa de desenvolvimento que conta com a pilha do protocolo TCP/IP integrada, permitindo implementar o acesso às redes WiFi por conter o *chip* ESP8266.

O segundo dispositivo foi o Raspberry Pi 3, que é um sistema embarcado baseado em um microprocessador capaz de fazer operações mais complexas que outros sistemas embarcados convencionais baseados em microcontroladores. O sistema conta com um processador Quad Core 1.2 GHz Broadcom BCM2837 64bit CPU e 1 GB de RAM, além de possuir interface para conexões sem fio como WiFi e Bluetooth (RASPBERRY, 2021).

Para os experimentos, foi utilizado também um módulo GPS do modelo Neo 6M para coletar as coordenadas utilizadas como carga para as requisições. Os dois sistemas embarcados podem ser vistos na Fig. 1.

Figura 1 – NodeMCU e Raspberry Pi 3 Model B usados nos experimentos.



Fonte – Autoria própria (2021).

2.3 wolfSSL

A wolfSSL é um conjunto de ferramentas criptográficas desenvolvidas principalmente para sistemas com poucos recursos, como os sistemas embarcados. A biblioteca é escrita em linguagem C com o propósito de ser leve e portável. Mesmo sendo uma biblioteca leve e rápida, possui suporte para a maioria dos protocolos de segurança e algoritmos criptográficos, como o TLS 1.3 utilizado nesta pesquisa. Devido a isto, é comum ser utilizada também em sistemas convencionais.

No Raspberry Pi a biblioteca foi instalada como em um sistema convencional. Por conter mais recursos que o nodeMCU, todos os recursos que são habilitados por padrão foram instalados na placa. Já no nodeMCU,

diversas modificações foram necessárias para que fosse possível rodar a aplicação. Foi necessário desabilitar as outras versões do TLS, deixando apenas a 1.3, desabilitar o RSA, habilitar as opções de otimização de código e memória e deixar apenas as cifras suportadas pelo TLS. O arquivo de configuração para o nodeMCU está disponível no repositório do *GitHub* em https://github.com/oaugustocezar/TLSESP8266.

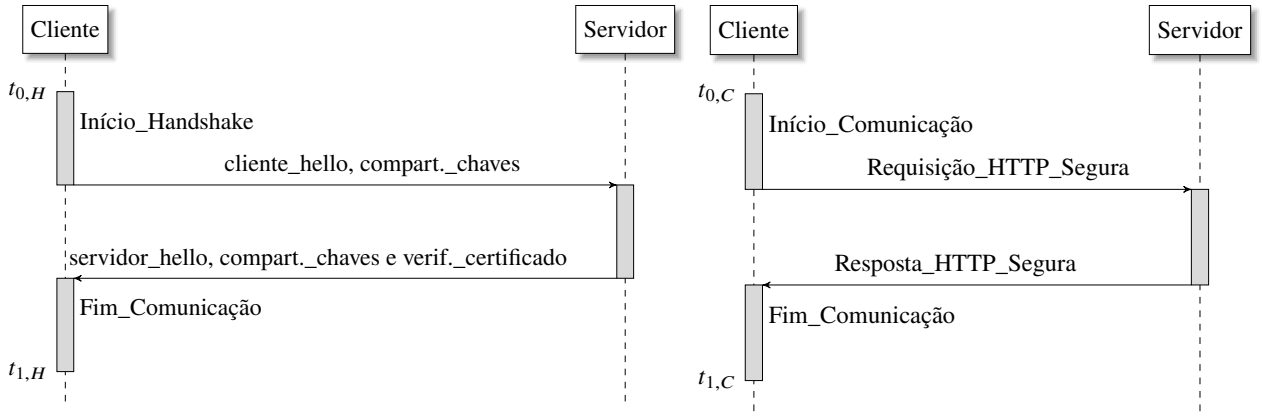
2.4 Experimentos

O servidor conta com três cifras, sendo elas: TLS_AES_128_GCM_SHA256, TLS_AES_256_GCM_SHA384 e TLS_CHACHA20_POLY1305_SHA256. Sendo as duas primeiras compostas por um algoritmos de criptografia autenticada que no caso é o AES GCM e um *hash* para garantir a integridade da mensagem. Nestes dois primeiros casos, há diferentes tamanhos de chaves e *hashes*, o que impacta no tempo de execução e no nível de segurança aplicado. Para a terceira conjunto de cifras citadas usa-se o ChaCha20 como algoritmo de criptografia de fluxo para prover a confidencialidade e o Poly1305 que é um autenticador de mensagem para fornecer autenticidade, além do *hash* para integridade.

Os dados transmitidos na comunicação possuem sempre um tamanho fixo, coletados por um sensor GPS acoplado aos sistemas embarcados, sendo estes dados a longitude, longitude e velocidade do dispositivo. Os tempos da comunicação e do *handshake* foram coletados nos clientes pelas aplicações desenvolvidas com o uso da wolfSSL, conforme ilustrado em Fig. 2a e Fig. 2b.

Figura 2 – Diagrama da comunicação segura com o TLS 1.3 entre cliente e servidor com coleta dos tempos.

(a) *Handshake*. (b) Troca de dados.



Fonte – Autoria própria (2021).

Os tempos do *handshake* e da comunicação foram calculadas conforme as **??** e **??**. Com os tempos coletados, as médias dos valores para cada cifra foram calculadas, assim como o desvio padrão e o intervalo de confiança para um nível de confiança de 95%.

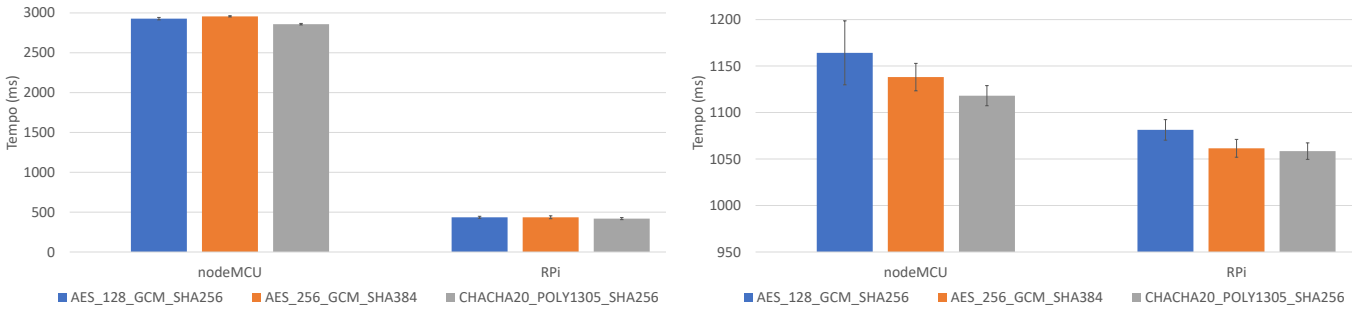
3 RESULTADOS

Os resultados dos experimentos podem ser vistos nas Fig. 3a e Fig. 3b para o tempo do *handshake* e da comunicação, respectivamente. Pode-se observar que os tempos coletados no *handshake* são praticamente iguais para todas as cifras, com um aumento de 591% para o nodeMCU, o que é esperado já que há uma diferença

considerável de processamento e memória entre os dois sistemas.

Figura 3 – Média dos tempos coletados na comunicação segura entre os sistemas embarcados e o servidor.

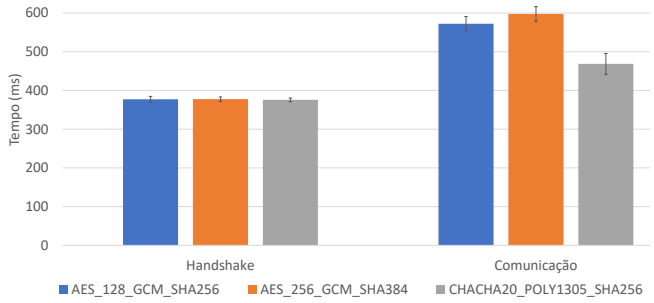
(a) *Handshake*. (b) Troca de dados.



Fonte – Autoria própria (2021).

Comparando as cifras na comunicação, o tempo varia de acordo com o algoritmo utilizado e com as suas chaves, sendo geralmente menor para a cifra baseada no ChaCha20. Porém, quando comparadas as cifras GCM, o resultado contradiz a teoria, pois o tempo medido para o AES com a chave de 128 bits foi maior do que para o AES com a chave de 256 bits. Conforme FERREIRA e SILVA (2020), o tempo de criptografia aumenta conforme o tamanho da chave aumenta. Para verificar possíveis erros na implementação do TLS nos servidores da Cloudflare, foi aplicado o TLS em um servidor local e executado novamente os experimentos, obtendo os resultados apresentados na Fig. 4, que se assemelham ao esperado.

Figura 4 – Tempo de *handshake* e de troca de dados coletados em um servidor local com o TLS 1.3.



Fonte – Autoria própria (2021).

Utilizando todos os dados, pode-se observar que o tempo das cifras que contém o AES-256 é maior do que as duas outras suítes, enquanto a que utiliza o ChaCha20 possui um tempo menor, provendo assim um impacto menor para a comunicação segura de sistemas embarcados.

4 CONCLUSÕES

Neste trabalho foram avaliados os tempos de *handshake* e de comunicação entre um servidor e dois sistemas embarcados, o Raspberry Pi 3 e o nodeMCU, utilizando o TLS 1.3 para prover confidencialidade, autenticidade e integridade dos dados. A arquitetura, como esperado, provou ser uma variável de grande influência no tempo da comunicação e isto se deve ao fato da diferença dos recursos de memória e processamento dos dois sistemas

embarcados, porém é viável a utilização do protocolo mesmo para sistemas baseados em microcontroladores, como o nodeMCU. Para a cifra AES_128_GCM_SHA256, como esperado, o tempo foi menor do que para a AES_256_GCM_SHA384, devido ao tamanho da chave e o tamanho do *hash* utilizado ser menor. Mas a cifra que obteve melhor desempenho foi a CHACHA20_POLY1305_SHA256, que é cerca de 22% mais rápida que o AES_128_GCM_SHA256 e 28% mais rápida que AES_256_GCM_SHA384.

Como contribuição, além das comparações dos tempos nos dispositivos, foi possível implementar e documentar os passos e configurações para executar o TLS 1.3 no nodeMCU, que até então, pouco foi documentado sobre. O erro encontrado na implementação do TLS 1.3 provido pela Cloudflare, foi devidamente reportado à organização e está em estudos para verificar os motivos do aumento do tempo para a cifra AES_128_GCM_SHA256.

AGRADECIMENTOS

Os autores agradecem à Fundação Araucária e à UTFPR pelo financiamento dessa pesquisa.

REFERÊNCIAS

ARFAOUI, Ghada et al. The privacy of the TLS 1.3 protocol. **Proceedings on Privacy Enhancing Technologies**, v. 2019, p. 190–210, 2019.

BANKER, Kyle et al. **MongoDB in Action: Covers MongoDB version 3.0**. [S.l.]: Simon e Schuster, 2016.

BARR, Michael. **Programming embedded systems in C and C++**. [S.l.]: "O'Reilly Media, Inc.", 1999.

DATTA, Parul; SHARMA, Bhisham. A survey on IoT architectures, protocols, security and smart city based applications. In: 2017 8th International Conference on Computing, Communication and Networking Technologies (ICCCNT). [S.l.: s.n.], 2017. P. 1–5. DOI: 10.1109/ICCCNT.2017.8203943.

DEWI, Estri JH; RUSYDI, Umar; IMAM, Riadi. **Implementation of Cloudflare Hosting for Speeds and Protection on The Website**. 2019. Tese (Doutorado) – Universitas Ahmad Dahlan.

FERREIRA, A. C.; SILVA, N. B. F. Comparison of secure communication with AES between embedded system and general purpose computer. In: 2020 IEEE Symposium on Computers and Communications (ISCC). [S.l.: s.n.], 2020. P. 1–6. DOI: 10.1109/ISCC50000.2020.9219672.

HAMEED, Ali; ALOMARY, Alauddin. Security Issues in IoT: A Survey. In: 2019 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT). [S.l.: s.n.], 2019. P. 1–5. DOI: 10.1109/3ICT.2019.8910320.

HANJURA, Anubhav. **Heroku Cloud Application Development**. [S.l.]: Packt Publishing Ltd, 2014.

HOWS, David; MEMBREY, Peter; PLUGGE, Eelco. **Introdução ao MongoDB**. [S.l.]: Novatec Editora, 2019.

RASPBERRY. **Raspberry Pi 3 Model B**. [S.l.: s.n.], 2021. Disponível em: . Acesso em: 20 ago. 2021.

SILVA, Bruno et al. Segurança de software em sistemas embarcados: Ataques & defesas. **Minicursos do XIII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais–SBSeg**, v. 2013, p. 101–155, 2013.

STALLINGS, William et al. **Computer security: principles and practice**. [S.l.]: Pearson Education Upper Saddle River, NJ, USA, 2012.