Comunicação segura em VANET

João L. M. Freitas, Leonardo da R. Souza, Patrick R. Sardou, Nilson M. Lazarin

[1]Bacharelado em Sistemas de Informação – Centro Federal de Educação Tecnológica Celso Suckow da Fonseca (Cefet/RJ) – Nova Friburgo, RJ – Brazil

{joao.frhb,leorsouza15,ptrcksardou,nilsonmori}@gmail.com

Abstract. *In the context of VANET networks, especially considering their applicability, it is of fundamental importance that the communication between the devices be carried out in a safe environment and without the interference of devices external to this communication. This article presents the implementation of secure communication through RF modules, using a cryptography library and a message exchange library. Procedures were carried out for the joint operation of these libraries and tests to assess the compliance of the encrypted communication.*

Resumo. *No contexto das redes VANETs, sobretudo considerando suas aplicabilidades, é de fundamental importância a comunicação entre os dispositivos ser realizada em um meio seguro e sem a interferência de dispositivos externos à essa comunicação. Este artigo apresenta a implementação de uma comunicação segura através de módulos RF, utilizando uma biblioteca de criptografia e uma biblioteca para troca de mensagens. Foram realizados procedimentos para a operação conjunta dessas bibliotecas e testes para avaliar a conformidade da comunicação criptografada.*

1. Introdução

As redes ad-hoc veiculares (VANETs) integram protocolos de conectividade móvel para agilizar a transferência de dados entre veículos bem como equipamentos em estradas. Na VANET, o dispositivo sem fio envia informações para veículos próximos e as mensagens podem ser transmitidas de um veículo para outro. Portanto, o uso da VANET pode aumentar a segurança e a otimização do tráfego. Semelhante a outras tecnologias, na VANET existem alguns problemas importantes e perceptíveis. Um dos mais importantes deles é a segurança. Uma vez que a rede é aberta e acessível de qualquer lugar no alcance do rádio VANET, a interferência nessa comunicação torna-se um alvo fácil para usuários mal-intencionados [Sabahi 2011].

Tendo em vista que as VANETs possuem, como uma das principais finalidades, obter informações em tempo real na comunicação entre veículos aplicando-as em medidas para aumentar a segurança no trânsito, tornar o compartilhamento dessas informações um meio seguro e sem a interferência de agentes externos, é imprescindível.

Este trabalho tem por objetivo tornar segura a troca de mensagens em uma rede VANET, realizando assim a criptografia na transmissão de mensagens entre módulos RF (Rádio Frequência) através da integração das bibliotecas Securino[1], uma biblioteca

[1]*https://github.com/nilsonmori/securino*

de segurança que atende as especificações do AES (*Advanced Encryption Standard*) e que implementa o modo de operação CBC (*Cypher Block Chaining*) com vetor de inicialização aleatório [Rocha et al. 2020] e a biblioteca Javino[2] que implementa um protocolo de comunicação para troca de mensagens entre a linguagem de programação Java e o Arduino através de uma porta serial [Lazarin and Pantoja 2015], estendida para permitir troca de mensagens *broadcast*, *multicast* e *unicast* em uma rede ad-hoc através de RF [Lazarin et al. 2021].

2. Fundamentação Teórica

Uma comunicação que ocorre em um determinado meio e onde há transferência de dados, pode ser considerada segura levando-se em conta três principais pontos: a preservação da confidencialidade, a integridade e a disponibilidade da informação [Kim and Solomon 2014]. A aplicação de técnicas criptográficas torna-se então uma solução comum para a obtenção e garantia de sigilo na troca de informações. Dada a facilidade de implementação e melhor desempenho é comum a utilização de criptografia simétrica em determinados contextos. Os principais modos de operação, nessa abordagem, são o ECB (*Electronic Codebook*) e o CBC.

O modo de operação ECB é um modo de confidencialidade que apresenta, para uma determinada chave, a atribuição de um bloco de texto cifrado fixo para cada bloco de texto simples. No modo ECB, sob uma determinada chave, qualquer bloco de texto simples sempre é criptografado para o mesmo bloco de texto cifrado [Dworkin 2001].

O modo de operação CBC é um modo de confidencialidade cujo processo de criptografia apresenta a combinação ("encadeamento") dos blocos de texto simples com os blocos de texto cifrado anteriores. O modo CBC requer um vetor de inicialização (IV) para combinar com o primeiro bloco de texto simples [Dworkin 2001]. Em seu processo de criptografia, o CBC pode operar com o vetor de inicialização de duas formas, mantendo-o fixo para a cifragem de cada bloco da mensagem, ou obtendo esse vetor de inicialização de modo aleatório.

Com vetor de inicialização fixo, a aplicação da cifragem em blocos que possuam o mesmo conteúdo, sempre resulta em um texto cifrado idêntico, facilitando assim a identificação de padrões a partir do texto cifrado por parte de um eventual atacante. Com a utilização do vetor de inicialização aleatório, mesmo que seja aplicada uma mesma chave de cifragem, em mensagens que possuam o mesmo conteúdo, o texto cifrado resultante é diferente, dificultando-se assim, substancialmente, o trabalho do atacante ao tentar identificar padrões de formação das mensagens através do texto cifrado das mesmas [Serafim 2012].

Na Figura 1 podemos observar um cenário onde um veículo recebe comandos remotos de uma base. Neste ambiente os dados podem ser capturados por um intruso e uma vez que utilizando ECB ou CBC com IV fixo a mensagem cifrada será também fixa. Na parte inferior da imagem podemos observar que utilizando o modo de operação CBC com IV aleatórios, um mesmo comando gera mensagem cifrada diferente a cada execução aumentando o nível de segurança.

[2] *https://github.com/profpantoja/javino-framework*

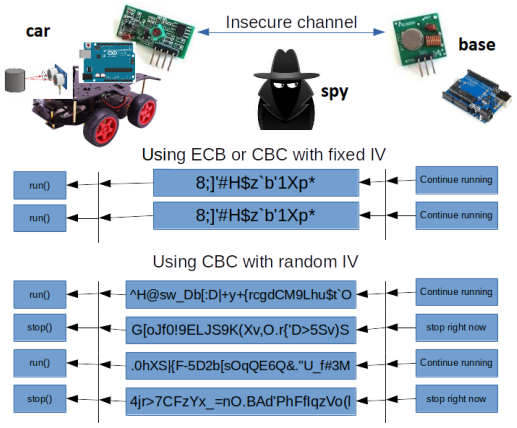


Figura 1. Comparação do uso de CBC com IV fixo ou aleatório.

3. Trabalhos Relacionados

A biblioteca VirtualWire[3] é uma biblioteca de comunicações para Arduino que permite a troca de mensagens via RF, através da utilização de transmissores e receptores RF de baixo custo [McCauley 2013]. Entretanto, ela fornece apenas comunicação via *broadcast*, sem endereçamento, confirmação de entrega ou sigilo.

O estudo realizado por [Rocha et al. 2020] sobre o desempenho e conformidade das bibliotecas de criptografia para aplicação na internet das coisas, apresenta um comparativo entre bibliotecas criptográficas para o Arduino. Os resultados obtidos demonstram que a biblioteca Securino se mostra superior às outras analisadas, por atender à todas as especificações do AES (*NIST FIPS PUB 197* e *SP 800-38-A*), garantindo um alto nível de segurança. Entretanto, a biblioteca apresentada não possibilita a comunicação RF.

O protocolo de comunicação apresentado por [Lazarin et al. 2021] possibilita que agentes BDI embarcados possam se comunicar como outros agentes embarcados de um SMA distinto de forma efetiva, através de RF, com mensagens *unicast*, *multicast* e *broadcast*. Entretanto o protocolo não implementa segurança na comunicação, trafegando as informações em claro.

Este trabalho apresenta a biblioteca Security-Vanets [4], através da integração das bibliotecas [Lazarin and Pantoja 2015], [McCauley 2013] e de [Rocha et al. 2020], possibilitando comunicação RF criptografada para redes ad-hoc veiculares, através da plataforma Arduino.

4. Implementação

A comunicação por Rádio Frequência ocorre por meio da transceptação de informações em sinais eletromagnéticos propagados no espaço. Para que esta comunicação ocorra, são necessários um dispositivo transmissor e um dispositivo receptor. Para a implementação do projeto foram utilizadas duas placas Arduino Mega, com dois módulos RF 433MHz, sendo um o transmissor e um o receptor. As placas Arduino foram conectadas a um computador via cabo USB e utilizamos a plataforma Arduino IDE para a compilação e execução dos códigos feitos.

[3] *http://www.airspayce.com/mikem/arduino/VirtualWire/*
[4] *https://github.com/joaofreittas/security-vanets*

A biblioteca Securino é a responsável por realizar a criptografia da mensagem que será enviada. Esta biblioteca, trabalha com um vetor de bytes hexadecimal. Foi necessária a criação de uma biblioteca adicional para a conversão de texto puro (*plain text*) para hexadecimal e vice-versa, nomeada Konverter. Após a mensagem estar devidamente criptografada, devemos utilizar a biblioteca Javino para o envio da mensagem através do módulo RF transmissor. Porém, haverá uma nova conversão pois o Javino trabalha apenas com codificação de texto base64. Dessa maneira, o vetor de bytes hexadecimal criptografado anteriormente será convertido para um texto com codificação base64. Para este fim, utilizamos a biblioteca Base64_Codec[5].

O módulo RF receptor recebe a mensagem criptografada e, de maneira contrária, integrado ao Securino, é realizada a decifragem da mensagem, obtendo assim a informação em claro que foi enviada. A Figura 2 ilustra o processo de envio e recebimento de uma mensagem através da biblioteca apresentada. A Figura 2a ilustra o tratamento da mensagem para transmissão. A Figura 2b ilustra o tratamento da mensagem durante a recepção.

(a) Fluxo de transmissão da mensagem. **(b) Fluxo de recepção da mensagem.**

Figura 2. Tratamento do dado para transcepção

5. Experimentos

Como prova de conceito, foi implementado um cenário contendo três membros (*Base*, *Espião* e *Veículo*). Cada membro da rede utilizou uma placa Arduino MEGA 2560 e um transceptor RF 433MHz.

- O membro *Base* realiza o envio de comandos *Andar* e *Parar* para o veículo, conforme programação apresentada na Figura 3a.
- O *Veículo* recebe e executa os comandos, conforme programação apresentada na Figura 3b.
- O *Espião* captura a mensagem no meio de difusão, conforme programação apresentada na Figura 3c.

Durante os teste de funcionamento o membro *Base* enviou apenas dois tipos de comandos, intercalados, um a cada segundo. O membro *Veículo* recebeu os comandos enviados, decifrou e executou a ação determinada. A troca de mensagens entre *Base* e *Veículo* é apresentada na Figura 4a. O membro *Espião* utilizou-se da biblioteca VirtualWire, apresentada em [McCauley 2013], para capturar todas as mensagens que trafegaram no meio de difusão.

[5] *https://github.com/dojyorin/arduino_base64*

```
void setup(){
  Serial.begin(9600);
  enableRF(12,11);
  setKey("0123456789abcdef");
}

void loop() {
  String cmd="command=go ahead";
  comm(cmd);
  delay(1000);
  cmd="command=stop now";
  comm(cmd);
  delay(1000);
}

void comm(String str){
  Serial.print("Send:");
  Serial.print(str);
  trasmiter(str);
  Serial.println("    [OK]");
}
```

(a) Código fonte da *Base*.

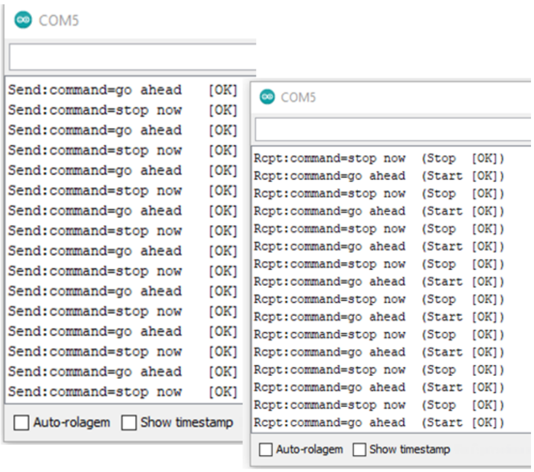
```
void setup() {
  Serial.begin(9600);
  enableRF(12,11);
  setKey("0123456789abcdef");
}

void loop() {
  if(receiver()){
    Serial.print("Rcpt:");
    Serial.print(getPlainTxt());
    action(getPlainTxt());
  }
}

void action(String act){
  if(act=="command=go ahead"){
    Serial.println("  (Start [OK])");
  }else if(act=="command=stop now"){
    Serial.println("  (Stop  [OK])");
  }else{
    Serial.println(" (CMD [ERROR])");
  }
}
```

(b) Código fonte do *Veículo*.

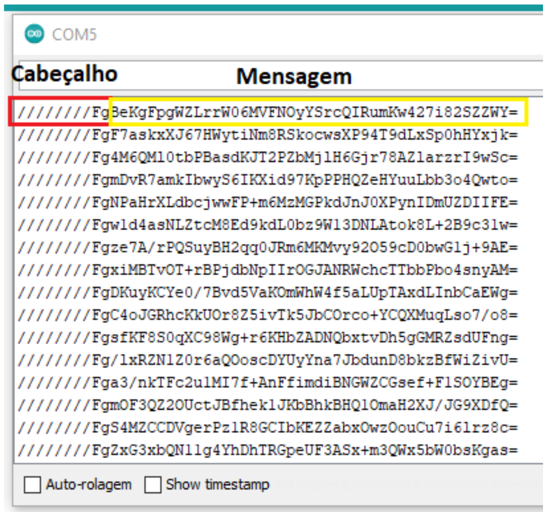
```
#include <VirtualWire.h>
void setup(){
    Serial.begin(9600);
    vw_set_rx_pin(11);
    vw_setup(2048);
    vw_rx_start();
}

void loop(){
    String m;
    uint8_t buf[VW_MAX_MESSAGE_LEN];
    uint8_t buflen = VW_MAX_MESSAGE_LEN;
    if (vw_get_message(buf, &buflen)){
      for (int i=0; i < buflen; i++){
        m=m+String((char)buf[i]);
    }
    Serial.println(m);
    }
}
```

(c) Código fonte do *Espião*.

Figura 3. Implementação do experimento.



(a) Troca de mensagens entre *Base* e *Veículo*.



(b) Mensagens interceptadas pelo *Espião*.

Figura 4. Experimento de transmissão e recepção de mensagens.

A Figura 4b apresenta as mensagens capturadas pelo *Espião*. A mensagem está formatada, conforme o protocolo definido em [Lazarin et al. 2021]. O cabeçalho da mensagem é dividido em três campos (*destino*, *origem* e *tamanho da mensagem*) representados em Base64. Vale ressaltar que a biblioteca fruto deste trabalho, implementa apenas comunicação *Broadcast*. Dessa forma, os enderenços origem e destino são *////* e o tamanho da mensagem é *Fg*, indicando que o campo mensagem possui 352bits, resultado de 128bits da mensagem original, somados aos 128bits do vetor de inicialização e aos 96bits oriundos da expansão sofrida na conversão para Alfabeto Base64.

Ao analisarmos o campo *Mensagem*, verificamos que o *Espião* capturou 16 mensagens distintas entre si, dificultando o relacionamento da mensagem cifrada com a ação realizada pelo veículo. A relação de uma mensagem em claro para várias mensagens cifradas é garantido pelo vetor de inicialização aleatório da biblioteca Securino, apresentada em [Rocha et al. 2020].

6. Conclusão

Este trabalho apresentou uma implementação de biblioteca de comunicação para VANET, através da evolução da biblioteca apresentada por [Lazarin and Pantoja 2015], permitindo comunicação *Broadcast* via Rádio Frequência, através da integração com a biblioteca apresentada por [McCauley 2013]. Além disso, foi possível garantir o sigilo da informação que transita no meio de difusão, através da integração com a biblioteca apresentada por [Rocha et al. 2020].

O modo de operação CBC com VI aleatório mostrou-se mais seguro, uma vez que a relação entre um mesmo comando enviado várias vezes, pelo membro *Base*, equivale à diversos textos cifrados que trafegaram pelo meio de comunicação. Dessa forma, acaba por dificultar a análise do *Espião*, na tentativa de identificação da relação do comando cifrado com a ação executada pelo *Veículo*.

Como trabalhos futuros pode-se considerar adoção completa do protocolo apresentado por [Lazarin et al. 2021], garantindo assim, o envio de mensagens *unicast*, *multicast* e *broadcast*, através do meio de difusão. Outra possibilidade de trabalho futuro é permitir o uso de tamanhos de texto e chave diferentes de 128bits, pois a biblioteca Securino implementa apenas o AES com chave de 128bits, bloco de texto fixo de 128bits e modos ECB e CBC com VI aleatório.

Referências

Dworkin, M. (2001). NIST Special Publication 800-38: Recommendation for Block Cipher Modes of Operation. *US National Institute of Standards and Technology*.

Kim, D. and Solomon, M. G. (2014). *Fundamentos de segurança de sistemas de informação*. LTC, Rio de Janeiro, 1a edition.

Lazarin, N. M. and Pantoja, C. E. (2015). A robotic-agent platform for embedding software agents using raspberry pi and arduino boards. *9th Software Agents, Environments and Applications School (WESAAC)*, pages 13–20.

Lazarin, N. M., Pantoja, C. E., and Jesus, V. S. d. (2021). Um Protocolo para Comunicação entre Sistemas Multi-Agentes Embarcados. *15th Workshop-School on Agents, Environments, and Applications (WESAAC)*.

McCauley, M. (2013). Documentation for the VirtualWire communications library for Arduino. Disponível em *http://www.airspayce.com/mikem/arduino/VirtualWire.pdf*.

Rocha, I., Schott, R., Verly, P., and Lazarin, N. (2020). Análise de desempenho e conformidade em bibliotecas criptográficas para internet das coisas. In *Anais da VI Escola Regional de Sistemas de Informação do Rio de Janeiro*, Porto Alegre, RS, Brasil. SBC.

Sabahi, F. (2011). The security of vehicular adhoc networks. In *2011 Third International Conference on Computational Intelligence, Communication Systems and Networks*, pages 338–342.

Serafim, V. d. S. (2012). Introdução à Criptografia: Cifras de Fluxo e Cifras de Bloco. Disponível em: *http://www.serafim.eti.br/academia/recursos/Roteiro_05-Cifras_de_Fluxo_e_Bloco.pdf*.