Universidade Federal do Espírito Santo
Centro Tecnológico
Programa de Pós Graduação em Informática

Internet das Coisas

Prof. Vinícius Fernandes Soares Mota

Laboratório Segurança

Este laboratório tem como objetivo apresentar os conceitos básicos de segurança. Uso de computação HASH com SHA-256 em ESP8266 e similares. Para este fim, você usará:

1. Um ESP8266
2. Um sensor de temperatura BMP280 (opcional e os dados poderão ser emulados)
3. Arduino IDE → Instalar a biblioteca ESP8266

O objetivo é aprender:

- Conceitos básicos de segurança em dispositivos embarcados
  - Assinatura digital
  - Algoritmos leves de criptografia.

Hardware

ESP8266, Sensor BMP280 (Opcional

Hashes

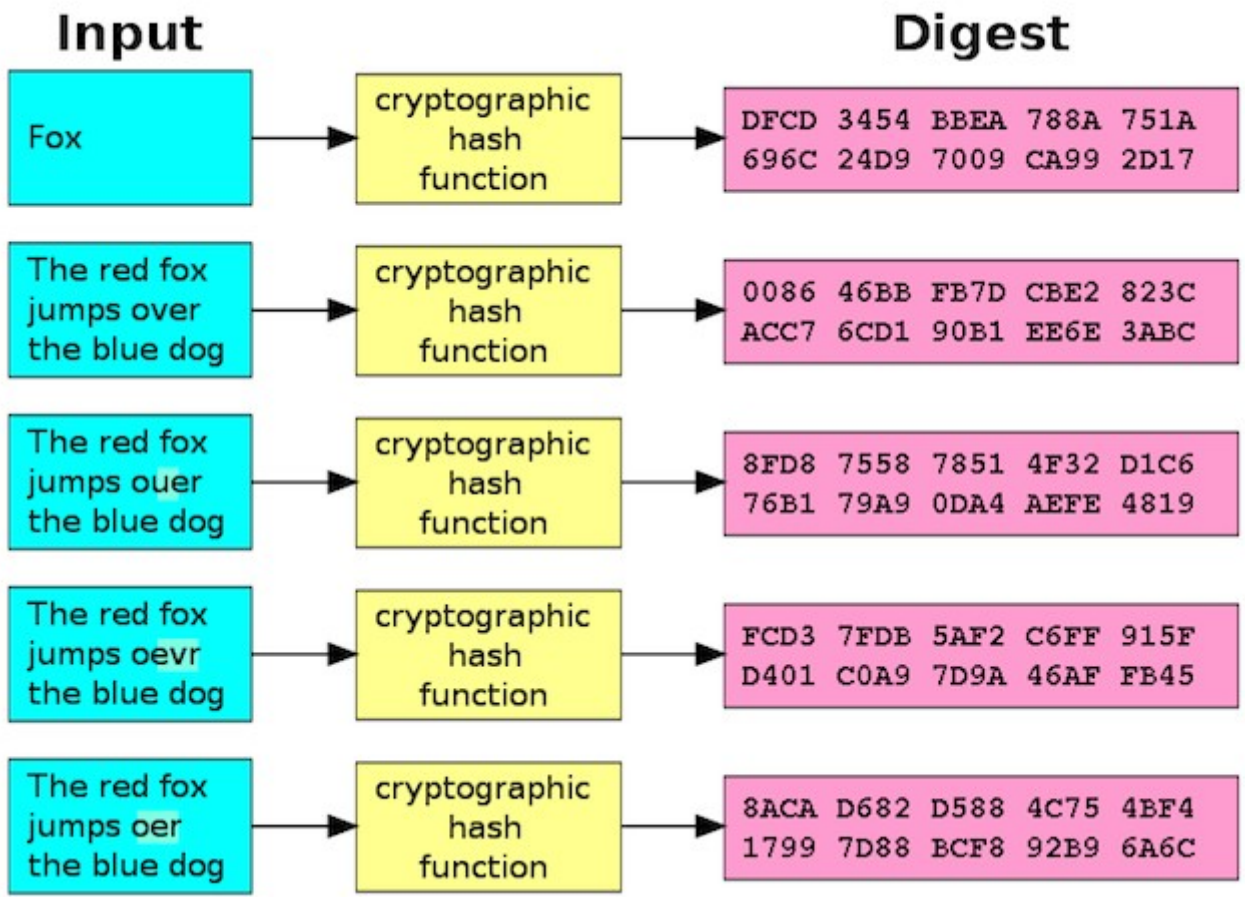
Uma hash criptográfico, chamado simplesmente como hash – é um algoritmo matemático que transforma qualquer bloco de dados em uma série de caracteres de comprimento determinado e fixo. **Independentemente do comprimento dos dados de entrada, o mesmo tipo de hash de saída será sempre um valor hash do mesmo comprimento.**

O objetivo de um Hash é que dados duas entradas diferentes, nunca se produza um hash igual, independentemente do tamanho da entrada. No caso do SHA1, todos os Hashes produzem 40 caracteres.

Os hashes criptográficos são utilizados para armazenamento seguro de senhas, garantir integridade de arquivos e mídias e para garantir integridade de uma mensagem.

Os mais comuns são o MD5, SHA3 e o SHA256. Segue o exemplo do SHA -1. Importante que para o SHA-1 já foram encontradas colisões, isto é, duas entradas distintas que geraram chaves iguais. Por isto ele não é considerado mais seguro.

Ver https://en.wikipedia.org/wiki/SHA-1



Exercício 1 - Para exemplificar, vamos utilzar o site https://codebeautify.org/sha1-hash-generator

“abc” gera como saída a9993e364706816aba3e25717850c26c9cd0d89d

Teste algumas entradas e veja a saída dos algoritmos SHA-1, SHA-3, SHA-256:

Exercício 2: Analisar o custo computacional de cálculo de HASHEs em ESP8266.

a. Instalar a biblioteca: https://github.com/rweather/arduinolibs
Documentação: https://rweather.github.io/arduinolibs/crypto.html

Essa biblioteca é a referenciada pela Arduino.ide
https://www.arduino.cc/reference/en/libraries/crypto/

Adicionar o diretório Crypto no diretório *home*/Arduino/libraries.

b. Executar e entender o código disponível no AVA, que foi baseado nos exemplos da biblioteca Crypto.

c. Para o próximo exercício, irá utilizar os exemplos da biblioteca Crypto:

- TestSHA256
- TestSHA512
- TestSHA3_256
- TestSHA3_512

Tabele e analise os resultados, apresentando qual o tempo por byte e por operação de cada modelo.

d. Uma forma de se garantir integridade de dados é adicionando um checksum aos dados enviados. Modifique o programa de coleta de temperatura para adicionar uma chave hash associado ao valor lido. Aqui você pode simular a temperatura utilizando a função random(min, max).

Algoritmos de criptografia

Algortimos de chave pública e privada

Um dos algoritmos clássicos de chave pública e privada é o algoritmo Diffie-Hellman.

Ver https://pt.wikipedia.org/wiki/Troca_de_chaves_de_Diffie%E2%80%93Hellman

Este algoritmo permite que se tenha uma “senha” pública e uma privada. Desta forma, usuários podem trocar mensagens privadas. Este é o princípio do conceito de chave pública e privada, no qual usuários que querem trocar mensagens compartilham apenas sua chave pública.

O algoritmo funciona da seguinte forma, primeiramente, define-se um número primo (p)e uma base (g) em comum, cada um define sua chave privada (a) a partir destes, calcula-se as chaves públicas, como CP = g^a mod p

O exemplo abaixo mostra como funcionaria a troca de chaves entre Bob e Alice, onde os valores públicos estão em azul e os valores secretos estão em **vermelho e destacados**:

- Alice e Bob entram em acordo para usar um número primo p=23 e como base g=5.
- Alice escolhe um inteiro secreto ***a*=6**, e então calcula sua chave pública e envia a Bob $A = g^a \bmod p$
  - $A = 5^6 \bmod 23$
  - $A = \mathbf{15.625} \bmod 23$
  - $A = 8$
- Bob escolhe um inteiro secreto ***b*=15**, e então calcula sua chave pública e envia a Alice $B = g^b \bmod p$
  - $B = 5^{15} \bmod 23$
  - $B = \mathbf{30.517.578.125} \bmod 23$
  - $B = 19$
- Alice calcula $\mathbf{s} = B^a \bmod p$
  - $\mathbf{s} = 19^6 \bmod 23$
  - $\mathbf{s} = \mathbf{47.045.881} \bmod 23$
  - $\mathbf{s} = \mathbf{2}$

Com a chave B de Bob, Alice calculou s e pode usar s para criptografar/descriptografar a mensagem.

- Bob calcula $\mathbf{s} = A^b \bmod p$
  - $\mathbf{s} = 8^{15} \bmod 23$
  - $\mathbf{s} = \mathbf{35.184.372.088.832} \bmod 23$
  - $\mathbf{s} = \mathbf{2}$

Com a chave A de Alice, Bob calculou s e pode usar *s* para criptografar a mensagem.

Uma implementação clássica do deste algoritmo aparece no Algoritmo Curve25519.
https://en.wikipedia.org/wiki/Curve25519

Exercício 3 - Analisar o tempo de execução do algoritmo Curve 22519 no ESP8266. Novamente, tabular e discutir os resultados. Isto é, execute pelo menos 3x o teste para

Bônus: Analisar os outros protocolos de segurança da biblioteca Crypto (AES, entre outros).