Lesson 9
Programming Arduino Hash, Encryption and Decryption Functions Usages (Examples 9.8 and 9.9)

Authentication and Encryption

- Authentication using a secret key and a hash function in place of communicating a plain text string
- Encryption using a secret key so that receiver can decrypt back the original string

Cryptosuite

- A library
- Downloadable from crypto website
- Encryption key length can be 128, 192 or 256
- Data encryption can use AES256 (Advanced Encryption Standard 256-bit encryption key length)
- DES (Data Encryption Standard)

Arduino AES 256 Library

- Downloadable and used for Arduino boards

Use of Hash Function

- Security adds by communicating hash of A1 or secret key (a text string) in place of communicating plain text string
- A function, called hash function creates a fixed length string, called hash of the input string, for example of A1
- The H(A1) from the original string, say A1. User communicates H(A1) in place of original A1.

Use of Encryption and Decryption functions

- Data needs protection from read and use by an in-between system. Encryption ensure the protection needs
- Uses of standard algorithms, for example, AES128, AES192, AES256 or DES
- Enable the encryption and decryption.

Examples 9.8 and 9.9

- Creation of authentication-code using Hash algorithm for secure communication of authentication codes
- Using SHA1 hash-function
- Use of the SHA1 at the application
- Example 9.9 explains the statements in C/C++ for using the AES256 encrypting and decrypting algorithms

Examples 9.8 and 9.9

- Step 1 Declaring the data types, constants, variables and functions used.
- Second and third steps are coding for setup() and loop().

Example 9.8 Step 1

- Step 1: Declare Header files inclusion and authcode data type
- /*Cryptographic library header files SHA1 or SHA256 or MD5*/ #include <sha1.h>
- /*IO utility functions*/ #include <util.h>
- /* Declare The authentication codes as set of unsigned integer of 8-bit */ uint8_t *authcode

Example 9.8

- Declare hashauthcode data type and assign an authcode
- /* The hash authentication codes declares as set of unsigned integer of 8-bit each at a pointed address*
- uint8_t *hashauthcode
- // Assign Values to authcode;
- .
- .

Step 2: setup ( )

- /* Initialising SHA1*/ sha1.init ( );
- /* create hash of authentication code, authCode*/ hashauthCode = sha1.result ( );
- /*Setup GPIO pin modes*/ pinMode (internalLED, OUTPUT); digitalWrite (internalLED, HIGH);
- /*Write statement for display on Serial Monitor of authentication code */ Serial.begin (9600);

loop ( ) Sender end statements

- /* Write Statements for communication of hash of the authentication code */

```
  ..
    test ( );
```

Example 9.8 Receiving End Statements

- Step 1: *#include <sha1.h>*
- #include <util.h>
- uint8_t *hashauthcode, *hashauthcodeNew
- Step 3: void loop () {
- /* Write Statements for receiving hash of the authentication code */
- ..

Example 9.8 Receiving End Statement

- /* Write statements for matching the hashauthCodeNew with stored hashauthcode*/
- /* Write statements for receiving the hashauthCode*/
- If (match_request = true) { if (hashauthCodeNew == hashauthCode) { mismatch = false;

Example 9.9

- Create Encryption of device sensed end messages
- Decryption at application end

Example 9.9 Statements

- Step 1: Preprocessor commands, declarations of data types and functions and include the required library files.
- /* First include the functions from Cryptographic library using pre-processor statement */

  #include <aes256.h>
- /*IO utility functions*/

   #include <util.h>

Example 9.9 Statements

- /* Contextual parameters Data type declaration */

aes256_context context /* Number of contextual parameters required that enables AES algorithm execute. These save at the pointed address context*/

Example 9.9 Step 2 setup ( )

- uint8_t key [ ] = {…, …., ….., ……, ……}/* Curly bracket has key of thirty two 8-bit unsigned integer numbers */

  aes256.init (&context, key);// initializes AES256

  char *message = "………"/* Assign Message characters for communication*/
- /*Write statement for display on Serial Monitor of authentication code */
- aes256_encrypt_ecb (&context, (uint8_t) message);

Example 9.9 Step 2 setup ( )

- /*Write statement for display on Serial Monitor of authentication code */

```
aes256_encrypt_ecb (&context, (uint8_t) message);
Serial.begin (9600);
```

- ..
- /* Write Statements for display of encrypted message */
- ..
- }

Step 3: loop ( )

- /* Write Statements for communication of encrypted message */
- ..
- test ( );

Step 1 at Application End Receiving Encrypted data

- #include <aes256.h> /* First include the functions from Cryptographic library using pre-processor statement */
- #include <util.h> /*IO utility functions*/
- *aes256_context context /* Number of contextual parameters required during* execution of AES algorithm. These save at the pointed address *context**/

Step 1 at Application End Receiving Encrypted data

- char *message = "………"// Assign Message characters for receiving the communication

Step 2: setup ( )

uint8_t key [ ] = {…, …., …..}/* Curly bracket has key of thirty two 8-bit unsigned integer numbers, each number separated by comma. */

aes256.init (&context, key) ;// initializes AES256

Serial.begin (9600);

- ..}

Step 3: loop ( )

- // Write statements for receiving the message for decryption ..

aes256_decrypt_ecb (&context, (uint8_t) message);

aes256_done (&context); / It ends the initialized AES256*/*

- /* Write Statements for display of decrypted message on serial monitor*/
- ..
- // Write statements for test ( ).
- …

Summary

We learnt

- The crypto-library functions enable the programming, secure communication of data for the IoT.
- Two security risks are taken care by (i) using a secret key and its secure communication using hass algorithm or message digest algorithm

Summary

We learnt

- (ii) using encryption and decryption functions, for examples, AES128, AES192, AES256 or DES

End of Lesson 9on
Programming Arduino Hash, Encryption and Decryption Functions Usages (Examples 9.8 and 9.9)