Hopping codes-based Authentication System

I. PROJECT DESCRIPTION

Design and implement an algorithm that performs rolling code approach to accomplish authentication between a transmitter and a reader. Both the transmitter and the reader use the same cryptographic pseudo-random number generator, so that each device produces the same sequence of unpredictable numbers for authentication. To keep the transmitter and the reader sequences synchronized. We will implement a cryptographic pseudo-random generator on the two end systems. We will achieved synchronization by using a shared initialization vector IV_i for each transmitter TX_i registered with the reader. The authentication protocol will keeps track of current IVs values, as these values will be updated during each transmitter-to-reader authentication process. Your system comprises of the following cryptographic subsystems

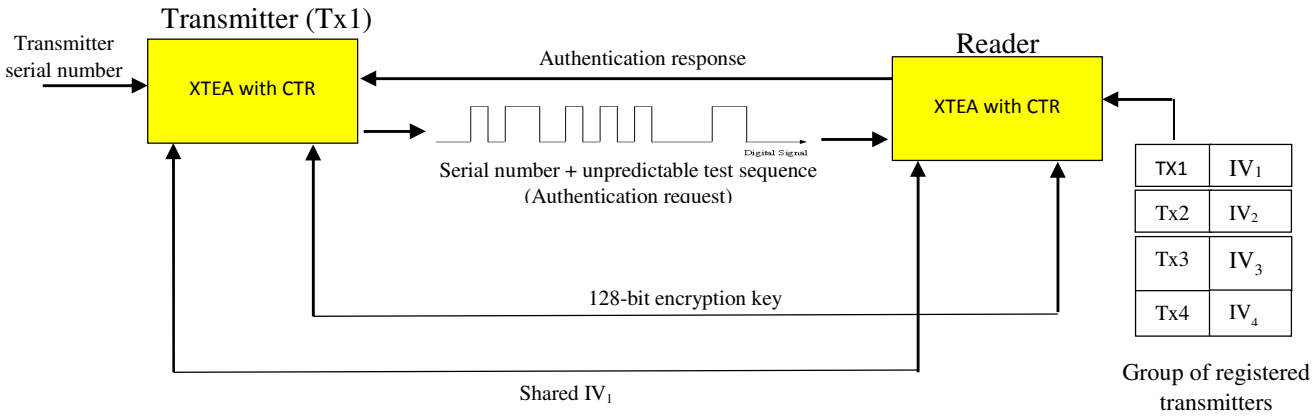

Figure 1: Hopping codes-based Authentication system

Database of Registered Transmitters **(10 points)**

The reader will be incorporated with a build in data base that stores a total of 10 transmitters' records. Each record in the database used to identify each registered transmitter and it contains two entries, one will hold the transmitter id for identification, and the second entry will hold the current value of the initialization vector for synchronization. During the authentication process, both entries will be used to compute the authentication response that will be rely back to the sender. Initialization vectors must be represented using 64-bits values. Meanwhile, each transmitter will be identified by 64-bits vector.

Authentication Request Packets **(10 points)**

Authentication request packets will have the following data format when transmitted over the channel. There will be a 64-bits field to hold the transmitter's id, a 16-bits field for the reader's id and a 64-bits that represent the unpredictable hopping sequence. The length of the packet is 144-bits as shown in figure 2.

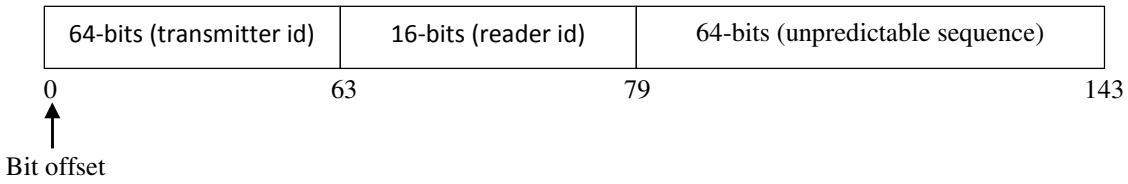

Figure 2: Authentication request packet

Authentication Respond packets **(10 points)**

Authentication response packets will be implemented using the following data structure. There will be a 64-bits field to hold the transmitter's id, a 16-bits field for the reader's id and an encrypted 64-bits value that represent the current transmitter's IV value plus 256 (IV+256). The length of the packet is 144-bits as shown in figure 3. Upon receiving the authentication request packet the reader will extract the transmitter id from the packet and accesses its database to determine the current IV value of the requester.

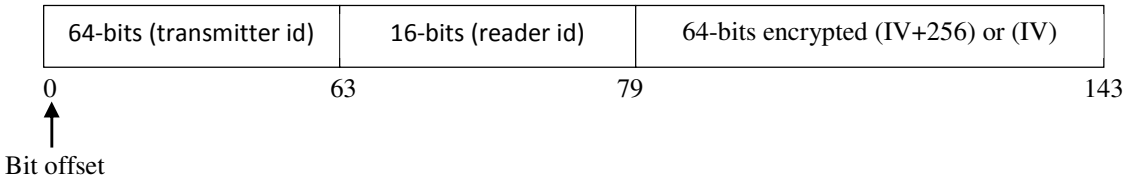

Figure 3: Authentication response packet

Authentication protocol **(20 points)**

At the transmitter, the unpredictable sequence is generated by applying CTR mode on the initialization vector IV as follows:

$$\text{Unpredictable sequence} = \text{transmitter id} \oplus \text{XTEA}((IV+i), K)$$

Where, $1 \leq i \leq 256$ and K is the encryption/decryption key. The transmitter will execute the CTR algorithm 256 times starting at $i = 0, 1, 2, \ldots, 256$ and generates 256 unpredictable sequences, then, it randomly chooses one sequence from the set which will be included within the authentication packet request and sent to the reader (see figure 4).

At the receiving end, the reader will compute a dataset of 256 unpredictable sequences by applying the same technique that is used by the transmitter. Then, the reader will search this dataset of unpredictable sequences to find one sequence that matches the received sequence. Upon a successful match, the reader will compute the next IV value of the requester, which is (IV+256), update the requester's record in the database to reflect the new changes, encrypt the value (IV+256) using XTEA, and includes it in the authentication response packet. In the case of no match, the reader will include the encrypted value of (IV) in the authentication response packet, without modifying the requester's internal record.

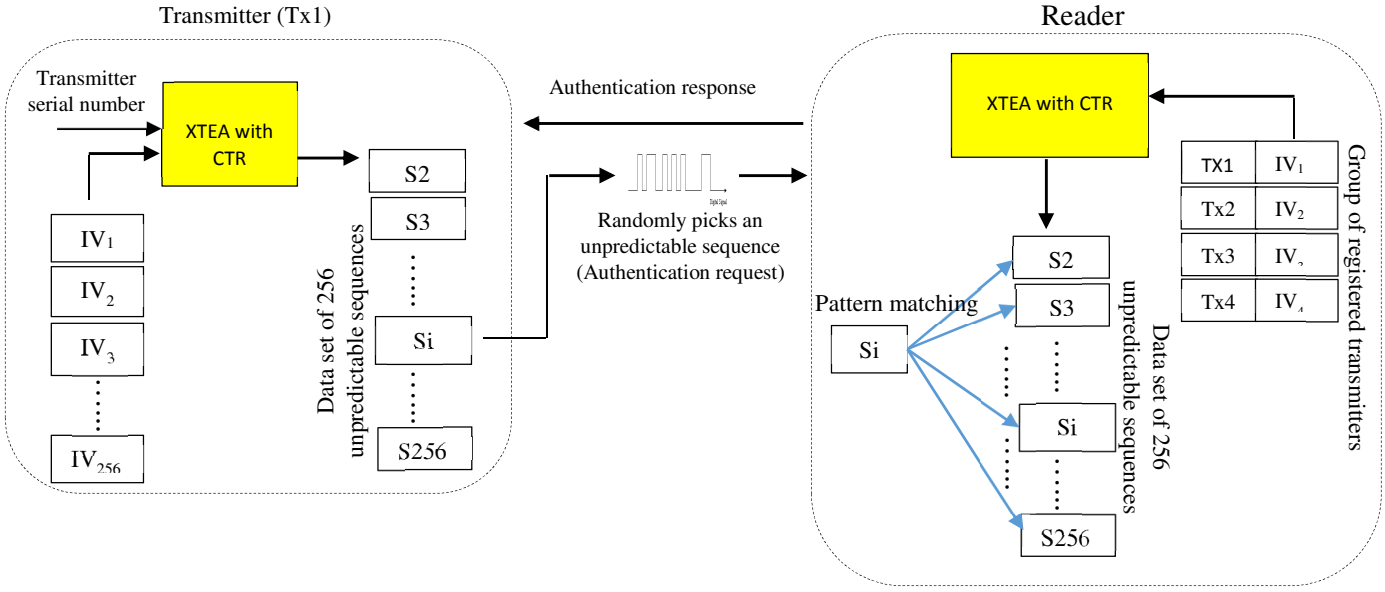

Figure 4: Authentication protocol

Encryption/Decryption Algorithms **(20 points)**

In this project, we will implement the X Tiny Encryption Algorithm (XTEA). XTEA is a block cipher that encrypt and decrypt a 64-bits block of plaintexts. The algorithm split the plaintext into two halves left and right. Each half contains 32-bits. XTEA uses a 128-bits key that is represented in the algorithm as array of 4 words (a word is 32-bits value). A constant called delta will be used during the encryption and the decryption process. A total of 64 rounds will be implemented for the algorithm. As dictated in figure 3, the following shows a block diagram of the XTEA.

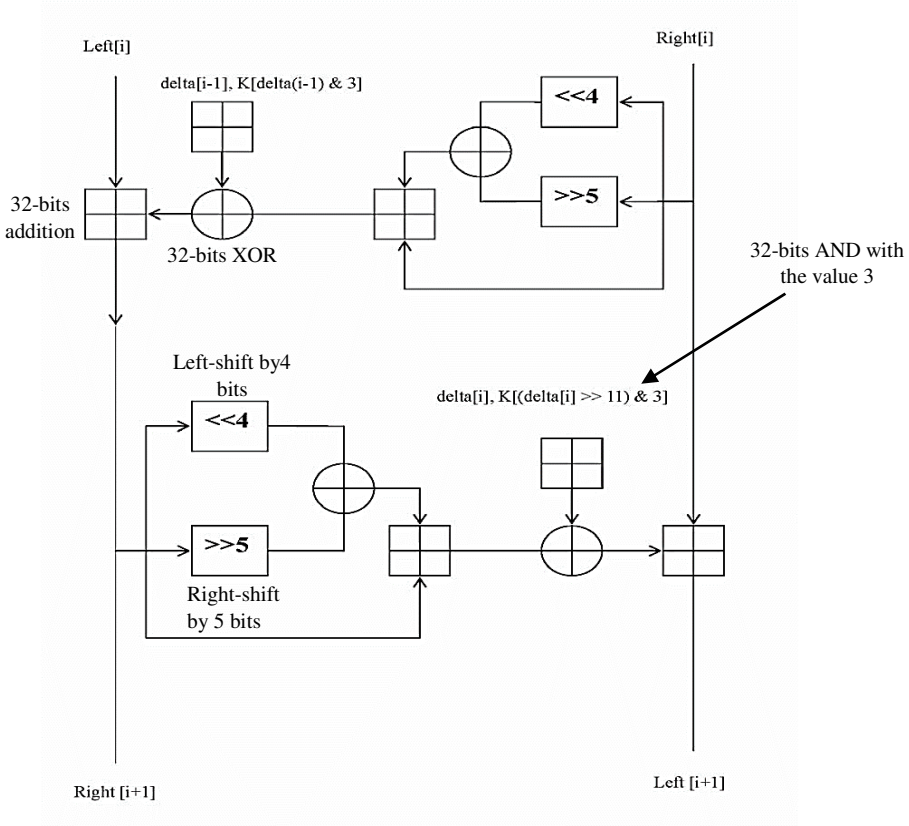

Figure 5: Abstraction of *i*-th cycle of XTEA

The relation between the output (Left[i+1], Right[i+1]) and the input (Left[i], Right[i]) for the ith cycle of XTEA is defined as follows:

$$\text{Left}[i+1] = \text{Left}[i] + F(\text{Right}[i], K[2i-1], \text{delta}[i-1]),$$
$$\text{Right}[i+1] = \text{Right}[i] + F(\text{Left}[i+1], K[2i], \text{delta}[i]),$$
$$\text{delta}[i] = (i+1)/2 * \text{delta}$$

The round function, F, is defined by

$$F(M, K[*], \text{delta}[**]) = ((M<<4) \oplus (M>>5))\ M \oplus \text{delta}[**] + K[*]$$

Where M is the Left and Right half in the algorithm. The key scheduling algorithm is implemented as follows

- Split the 128-bit key K into four 32-bit blocks
  K = (K[0], K[1], K[2], K[3]).
- For rounds r = 1,….,64
  if (r is odd) then
  key = K[delta((*r*-1)/2) & 3];
  else
  if (r is even) then
  key = K[(delta((*r*/2)>>11) & 3]

Note: All arithmetic operations including XOR are based on 32-bits processing.

II. PROJECT REPORT AND SUBMISSION REQUIREMENTS (20 points)

- Write and submit a final report that outline your design. Describe all the classes that you design and developed for this project.
- Each team will require to submit one executable application. Your application must include all the files, executable files, and header files.
- Submit a hardcopy of the user manual and the project report
- Encryption keys and Initialization vectors should be user input data.
- On the project due date. Each team will be asked to give a live demo during class time.
- No libraries or dependencies class are allowed during the design and the implementation for this works. All algorithms must be coded by the developers.

Reward points for your work:

- There will be a project contest among all teams. The team with the best GUI implementation will get **20 rewards points**.

Note*: The authenticity of your work will be evaluated. Copying materials/code from online resources is prohibited and will not be tolerated.*

III. PROJECT TEAMS:

- **Team 1**: Schmid Tosha, Bonsignori Evan, Brown Darryle, Schnibben David, Norris Tyler
- **Team 2**: Bradley Timothy, Komula Dillon, Devore Johnathan, Thomas John, Perez Wendy
- **Team 3**: Macdonald Franklin, Mazzolini Francis, Koba Alexander, Wilcox Cameron, Popov Brett
- **Team 4**: Moody Michael, Patel Dipiksha, Lartey-young Derek, Kelly John, Thornton Elena
- **Team 5**: Perry Michael, Harper Lazaryia, Patel Prashill, Mcintosh Joseph, Lingard Brooke

IV. GROUP MEETING (10 points)

Weekly meeting at the instructor office will be conducted with each group to monitor your progress.