CS478 Lab 3

By: Hayden Anderson, Courtney Miller, and Rex Henzie

a. The first main idea behind this algorithm is to shrink/compress data that needs to be sent, to save memory. The second main idea behind the algorithm is to make sure that data hasn't been deleted (won't stop it from being deleted, but you'd be able to detect it). The third main idea is that you can't modify or change the data without being detected. The next main idea is the data must be confidential (not just anyone can see it/decrypt it) and the same time the data's sender will be authenticated and the data's integrity is good (hasn't been corrupted). Another main idea is that the sensor doesn't have to compute all the individual authentication tags (at time of uploading data) nor does it have to store a large amount of authentication tags (it should be small amount required to store). The last idea is that the there must be resiliency to not all the packets arriving to the destination, due to either just chance (it happens) or some active jammer; this last idea means that 100% of the packets don't have to arrive for the receiver to be able to get all the data.
    - Compromise Resiliency - This is achieved by the sender algorithm, step 3 and step 6. The HMAC in step 3 will preserve integrity and authentication of the data, step 6 deletes the previous state (including cryptographic key, data item, and authentication tag). Step 3-5 of recovery makes sure nothing was modified and proves the authentication of the entity who sent it is the sensor.
    - All-or-Nothing Verification - Step 3-6 of of the sender algorithm will make sure that if an attacker deletes a piece of data then it will be detected because then in step 3-5 of the receiver's algorithm it will detect if $s'_{1,n}$ is equal to $s_{1,n}$ if there is any missing data then the $s'_{1,n}$ will not equal $s_{1,n}$, thus the deletion will be noticed.
    - Compression - Step 1 of sender's algorithm is used to compress the data, while step 7 of the receiver's algorithm will decompress it.
    - Aggregate Authentication - Step 3 - 7 of sender's algorithm. Step 3 -6 of the sender's algorithm will let you store only the most recent authentication tag while step 7 will allow this to happen (send the data and the most recent authentication tag $s_{1,n}$. Step 1 through step 5 of receiver's allows this to work by recursively going through and reconstructing each of the authentication keys (and of course making sure they are correct).
    - Packet Loss Resiliency - Step 7 of sender's algorithm uses Rabin Information Dispersal algorithm to add resiliency to jamming and packet loss. Step 1 and 6 recovers the message from Rabin Information Dispersal algorithm.
    - Authentication and Integrity - Step 2 of the sender's algorithm provides confidentiality through encryption, while step step 6 of receiver's algorithm decrypts it. Step 3 of the sender's algorithm provides authentication and integrity while step 3 of the receiver's algorithm makes sure this is enforced (if both do it,

then it must equal each other, thus proving that the sender is authenticated as the sensor due to it having the symmetric key), while integrity is double checked after this step in step 4 and 5 of the receiver's algorithm (all of them must equal each other) else the data could be different or corrupted.

b. Compression must be done before encryption because compression doesn't really work on "random" data. Encryption must be done before authentication or the security is broken. Authentication should be done before aggregation because you can't free up memory that has your current authentication tag until you use it, and make then next tag. Redundancy should come last and after aggregation because there is no point in using more memory until you're ready to send and it is only done at this time to save memory.