Andrew Sanchez

1717181

2/18/21

CSE-13

DESIGN PDF

Prelab Part 1

1. Pseudocode for insert and delete:

```
void bf_insert (BloomFilter *bf, char *oldspeak){
  bv_set_bit(bf->bv, hash(bf->primary, oldspeak));
  bv_set_bit(bf->bv, hash(bf->secondary, oldspeak));
  bv_set_bit(bf->bv, hash(bf->tertiary, oldspeak));
  return;
}

void bf_delete (BloomFilter *bf, char *oldspeak){
  bv_clr_bit(bf->bv, hash(bf->primary, oldspeak));
  bv_clr_bit(bf->bv, hash(bf->secondary, oldspeak));
  bv_clr_bit(bf->bv, hash(bf->tertiary, oldspeak));
  return;
}
```

Prelab Part 2

1. Pseudocode for linked list functions:

```
LinkedList *ll_create(bool mtf){
  *ll = malloc(sizeof(Linkedlist));
  ll-> mtf  = mtf;
  ll ->head = node create();
  ll ->tail = node create();
  ll ->length = 2;
  return ll;
}
```

```
void ll_delete(LinkedList **ll) {
  for(i = 0; i< ll->length) {
    node_delete(i)
  }
  free(*ll);
  *ll = NULL;
  return;
}

uint32_t ll_length(LinkedList *ll) {
  return ll->length;
}

Node *ll_lookup(LinkedList *ll, char *oldspeak) {
  For (i = 0; i < ll->length; i++ {
    If (ll[i] == oldspeak) {
      If(ll->mtf) {move_to_front(ll[i])}
      Return ll[i];
    }
  }
  Return NULL;
}

void ll_insert(LinkedList *ll, char *oldspeak, char *newspeak) {
  if (ll_lookup(ll oldspeak) == NULL) {return;}
  ll[head-1] = node create(oldspeak, newspeak);
  return;
}

void ll_print(LinkedList *ll) {
  for (i = 0; i < ll->length; i++) {
    node_print(ll[i]);
  }
  Return;
}
```

Prelab Part 3

1. The regular expression: # define WORD "[a-zA -Z_-”’]+"

Using the sorting information gathered from the prelab, The design of this firewall is to use the salts as a form of library for translation, and the hashmap will serve as the rosetta for the library of oldspeak and convert it to newspeak. The linked lists are a way of taking in oldspeak and returning whether to convert it to newspeak or deem it as badspeak based on the results of the hashing. This will require a number of structs and files to link all of this information together. At the very least I want to have all the major structs that I will be needing here:

```
Struct BloomFilter {
  Uint64_t primary[2];
  Uint64_t secondary[2];
  Uint64_t tertiary[2];
  BitVector *filter;
}; //Pg. 2-3

Struct BitVector {
  Uint32_t length;
  Uint8_t *vector;
}; //Pg. 4-5

Struct HashTable {
  uint64_t salt [2];
  uint32_t size;
  bool mtf;
  LinkedList ** lists;
}; //Pg. 6-7
```

```
Struct Node {
  char * oldspeak;
  char * newspeak;
  Node * next;
  Node * prev;
}; //Pg. 8

Struct LinkedList {
  uint32_t length;
  Node * head;
  Node * tail;
  bool mtf;
}; //Pg. 9-10
```