CS558 Notes

January 24th

Logistics

- Lab 1 released tonight (01/24)

Encryption

Encryption provides confidentiality by securing a message so no one other than the sender and intended recipient can read it.

Shift Cipher

Shift Cipher (from Julius Caesar; aka caesar cipher) provides encryption by replacing the letters of a message with letters corresponding to a certain number of letters up or down the alphabet.

It is one of the simplest forms of encryption, see below for an example how to decrypt it.

Example (*plaintext recovery attack*):

- **Encryption Protocol**: `Q` in ciphertext corresponds to `I` in plaintext
- **Message To Decrypt**: `BR ZKR KBR PBRRNR`

1. `R` **corresponds to** `E`: `BE ZKE KBE PBEENE`
2. `B` **corresponds to** `H`: `HE ZKE KHE PHEENE`
3. `K` **corresponds to** `T`: `HE ZTE THE PHEENE`
4. `Z` **corresponds to** `A`: `HE ATE THE PHEENE`
5. `P` **corresponds to** `C`: `HE ATE THE CHEENE`
6. `N` **corresponds to** `S`: `HE ATE THE CHEESE`

The problem with this protocol is that it's easy to break. *Frequency analysis* refers to inferring from the letters and their frequency, what the decrypted message might say. The more letters we have successfully decrypted, the more clear our message becomes. Already at step 3 in the above example we could make an educated guess at to what the original message contains.

To combat this, we should not make our protocol such that the same letters in ciphertext always correspond to the same letters in plaintext.

Symmetric Encryption

To create a encryption scheme, we must ensure that the encryption is symmetric. To do this, we need the following:

1. Correctness

An input always maps to a unique output, and the output always uniquely map back to the original input. If we have the key, we can always recover the original message (see MD5).

`Decypher( Encrypt( message )) = message`

2. Security

All encryption schemes must be secure even though the messages are not random.

There are several ways to break an encryption scheme, and a good one must be able to resist them all. There are three main ways to break an encryption:

1. **Indistinguishability attack**: given a cipher `c`, decide if it corresponds to a given message `m`.
2. **Plaintext recovery**: given a ciphertext, learn the original plaintext message.
3. **Key recovery**: learn the key used to encrypt our messages.

The One-Time Pad Encryption Scheme

Assuming we are to encrypt only one bit, a way to do this is to generate a random key of value either `1` or `0` with probability `0.5` for both. We can then `xor` our message (bit) with the key, and get an encrypted message that can only be decrypted if we know the key.

```
Encrypt( message ) = message xor key
Decrypt ( encrypted message ) = encrypted message xor key
```

One major drawback of this encryption scheme is that we must use a new key for every message, and as such it inefficient and not used in practice. Using the same key for two messages would easily allow the attacker to crack our key, and therefore decrypt every message we've sent using that key.

The one time pad (for one bit) satisfies our definition of security because the adversary (attacker) cannot determine our key, and therefore message, with better accuracy than 50% (by guessing which one of two values we've used). See below for a proof.

- Observe that `k`, `b` are random variables that takes on value `0` with probability `0.5` and value `1` with probability `0.5`.
- We have two unique messages `m0` and `m1`.
- `b` refers to the message we have chosen to encrypt.
- `c*` refers to the encrypted message of message `*` (unknown).
- Our encryption scheme is `m0 xor k = c*`.
- What is the probability that `encrypt(m0) = c*` equals the probability that `b = 0`?
  Probability of `m0 xor m0 xor k = c* xor m0` equals the probability that `k = m0 xor c*`.

  Conclusion: The adversary cannot do better than guessing the value of `k` with a 0.5 probability of guessing correctly.

MD5 Hashing

The MD5 protocol takes bit strings of arbitrary length and maps them to bit strings of length 256. Each always input maps to a unique output, but we only have `2^256` possible output bit strings, and therefore a single output may map to multiple inputs.

A MD5 hash function (or any hash function) cannot be an encryption scheme, becasue there is not a unique mapping from outputs to inputs (multiple messages can encrypt to the same ciphertext, and we may never recover the original message).

Authentication

Authentication provides integrity by making sure that the message came from the correct person and is unchanged.

- Message Authentication Code (MAC)
- Digital Signatures