Cryptography

David Shiroma & Sanjana Sathyanarayanan

Announcements

- Tomorrow's section:
  - Final review!
  - Last official sections :(
    - TA's Choice Thursday
- Simulated Final released
  - Key released (tentatively): Thursday 3/11
  - Due: Sunday 3/14
- HW8: Huffman released
  - Due: Friday 3/12
  - *No resubs
- *Last* round of resubmissions:
  - Due: Sunday 3/14
- Course Evaluations

Cryptography

- “The practice and study of techniques for **secure communication** in the presence of third parties called adversaries”
- Guaranteeing confidentiality of data
  - Encryption
- Guaranteeing integrity of data
  - Message authentication
- Proving identity
  - Digital signatures, certificates

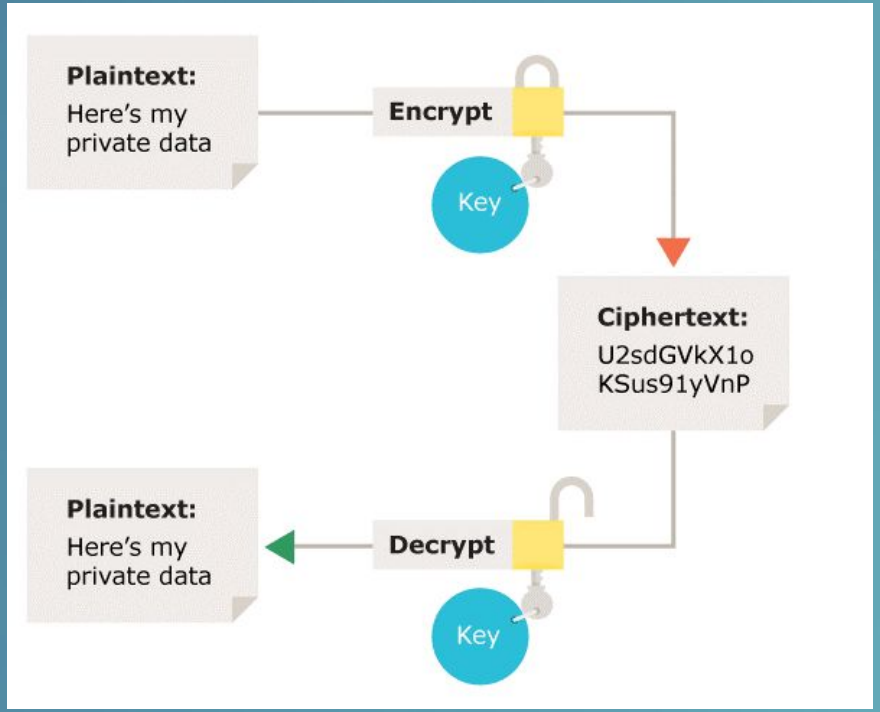
Encryption

- Encryption: The process of encoding information
  - Converting a plaintext into a ciphertext using an encryption algorithm
- Decryption:
  - Decoding the ciphertext into the plaintext
  - Typically reverses the encryption algorithm

Encryption

- What do we need in an encryption algorithm?
  - Complex
    - But not too expensive to compute
  - Reversibility
    - Easy for intended recipient to decrypt
    - Hard for attackers/outsiders to decrypt
      - **Keys**

Symmetric Encryption



Cipher shift

What is it?

A way to encode a message

How does it work?

Use a permutation of numbers to make a ciphertext out of plaintext

e.g. Substitution Cipher (Key = shift of 2)

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

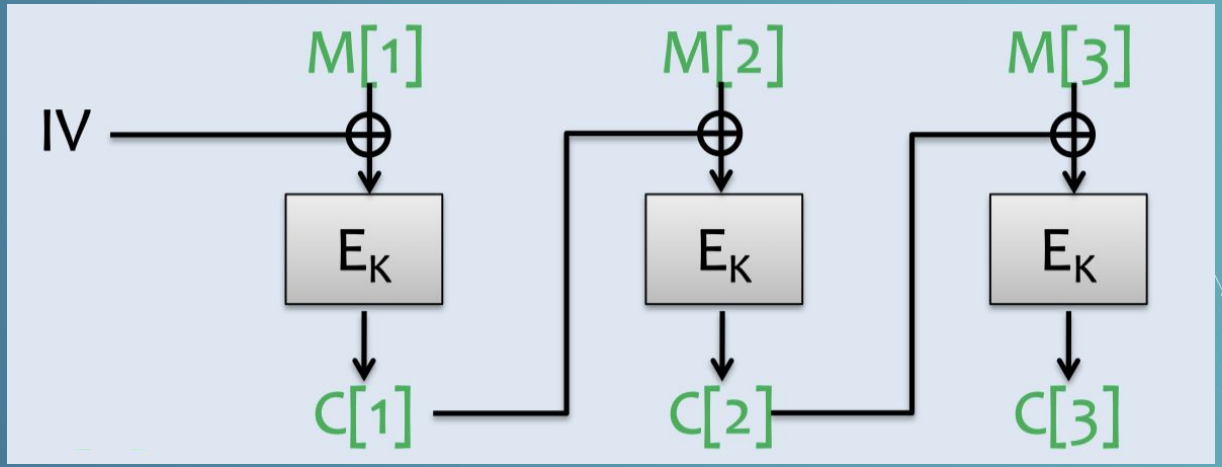
C D E F G H I J K L M N O P Q R S T U V W X Y Z A B

Plaintext: seattle

Ciphertext: ugcvvng

Ciphertext Block Chaining

- Chaining mechanism that causes the decryption of a block of ciphertext to depend on all the preceding ciphertext blocks
- Adds complexity

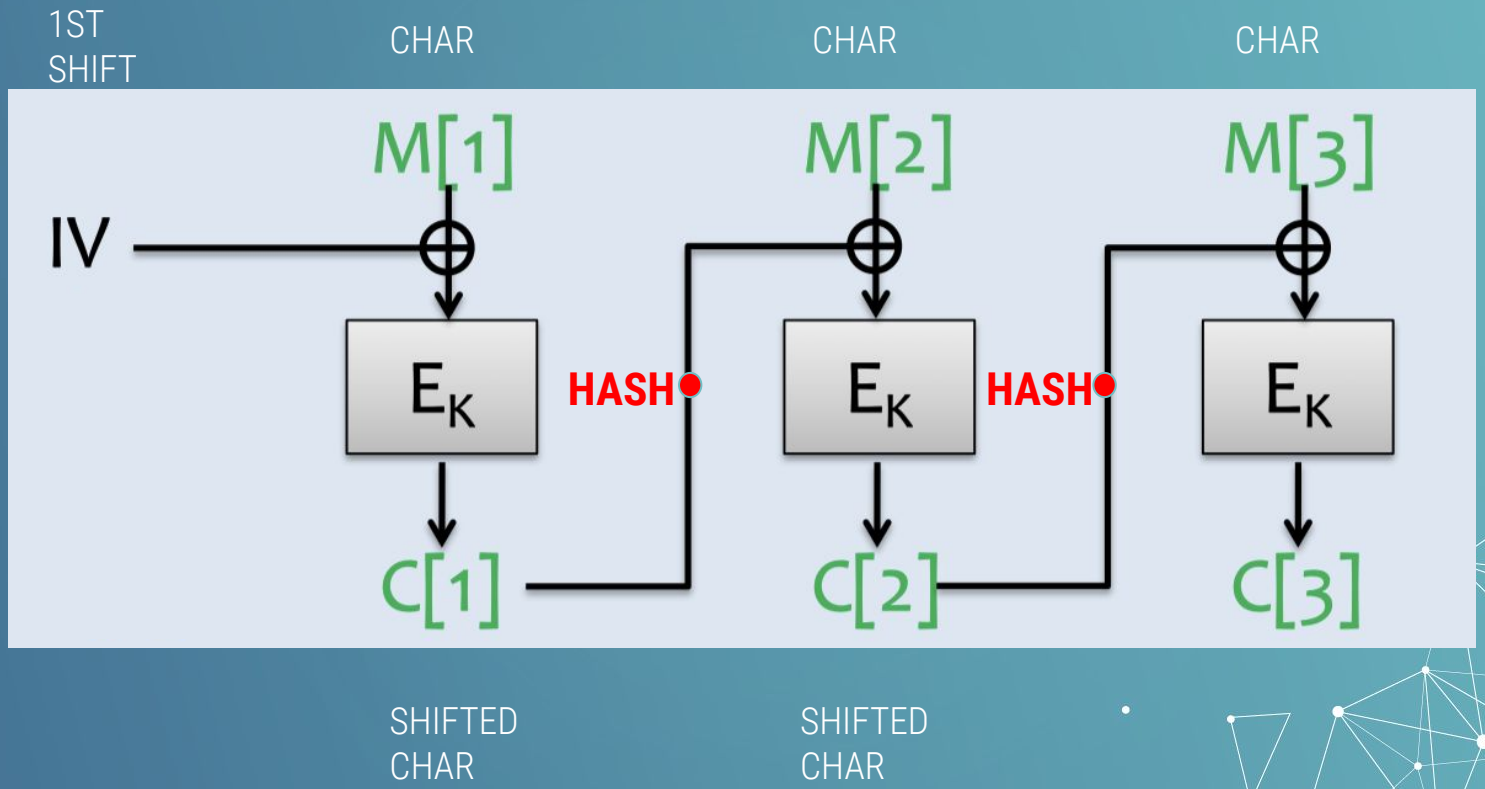


Applying it to our model? → Chain by letter

Ciphertext Block Chaining

- What can we do to the ciphertext and shift of the previous letter to make a shift for the next letter?
    - Need a function that can map ciphertexts to ints
    - Remind you of anything we recently learned?
        - → **Hashing**

Ciphertext Block Chaining



Ciphertext Block Chaining

Encrypt: “abc” with shift = 2

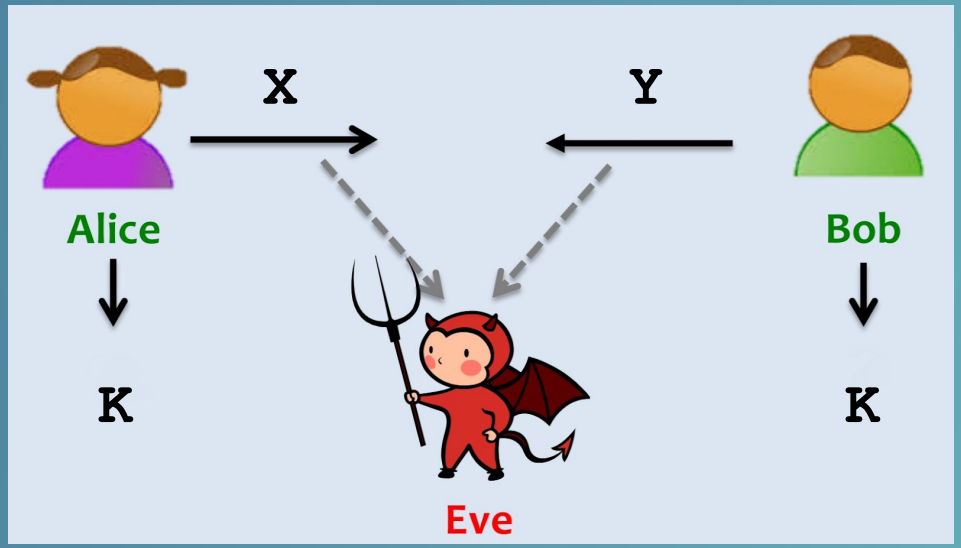
1. a → c
2. hash(c + 2) → 23
3. b → y
4. hash(y + 23) → 18
5. c → u

“abc” → “cyu”
Notice: “cyu” harder to decrypt than “cde”

Key Exchange

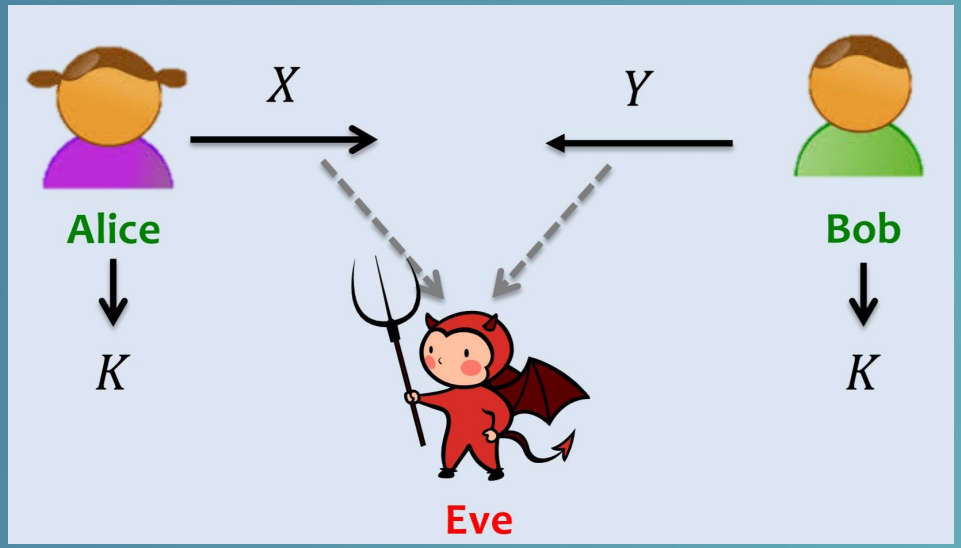
Key Exchange

- Alice and Bob start with nothing and agree on a key, secret and random to Eve
  - {X, Y, K} ≈ {X, Y, K'}: "After seeing X and Y, the agreed key K is pseudo-random to Eve"



Key Exchange

- Alice and Bob start with nothing and agree on a key, secret and random to Eve
  - {X, Y, K} ≈ {X, Y, K'}: "After seeing X and Y, the agreed key K is pseudo-random to Eve"



How do Alice and Bob actually compute X, Y, and K?

Necessary Background

- Recall (from algebra): for a real number X and base r, $\log_r X$ represents the exponent which I need to raise r to in order to get X; that is, $r^{(\log_r X)} = X$
- Recall (from 142): the mod (%) operator
- Observation: if I take all the powers of 3 and mod by 7 when necessary, I will eventually cycle through all positive remainders % 7 (excluding 0)

i	0	1	2	3	4	5	6	7	8
3^i	1	3	9	27	81	243	729	2187	6561
3^i % 7	1	3	2	6	4	5	1	3	2

Necessary Background

- Observation: if I take all the powers of 3 and mod by 7 when necessary, I will eventually cycle through all positive remainders % 7 (excluding 0)
- Discrete Logarithm Problem: given a base r and a some remainder X mod n, “hard” to find y s.t. r^y mod n == X (one-wayness)

i	0	1	2	3	4	5	6	7	8
3^i	1	3	9	27	81	243	729	2187	6561
3^i % 7	1	3	2	6	4	5	1	3	2

Diffie-Hellman Key Exchange

- Baked into protocol is the modulus (n) and base (r) that will be used for key exchange*

Alice:

```
Random rand = new Random();
int x = rand.nextInt(n);   3
int X = Math.pow(r, x) % n; 6

// send X to Bob

int Y = // receive Y from Bob 5

int key = Math.pow(Y, x) % n;
// (r^y)^x = r^(yx) = r^(xy)
```

5^3 % 7 --> 125 % 7 --> 6

Bob:

```
Random rand = new Random();
int y = rand.nextInt(n);   5
int Y = Math.pow(r, y) % n;  5

int X = // receive X from Alice 6

// send Y to Alice

int key = Math.pow(X, y) % n;
// (r^x)^y = r^(xy)
```

6^5 % 7 --> 7776 % 7 --> 6

Where to go from here

Questions that you might have thought of...

- When is an attack “successful” and when is a scheme “secure enough”?
  - security goal + threat model
- Is key-exchange a necessary part of an encryption scheme?
  - public key (a.k.a. asymmetric) cryptography
  - Neal Koblitz + (hyper)elliptic curve cryptography
- *Shor’s Algorithm?

CSE 490c - Cryptography (soon to be permanently numbered?)

Where do cryptographers go when they die?

En-crypts :)