ILLINOIS INSTITUTE OF TECHNOLOGY

CS 528 (Fall 2021)
Data Privacy & Security

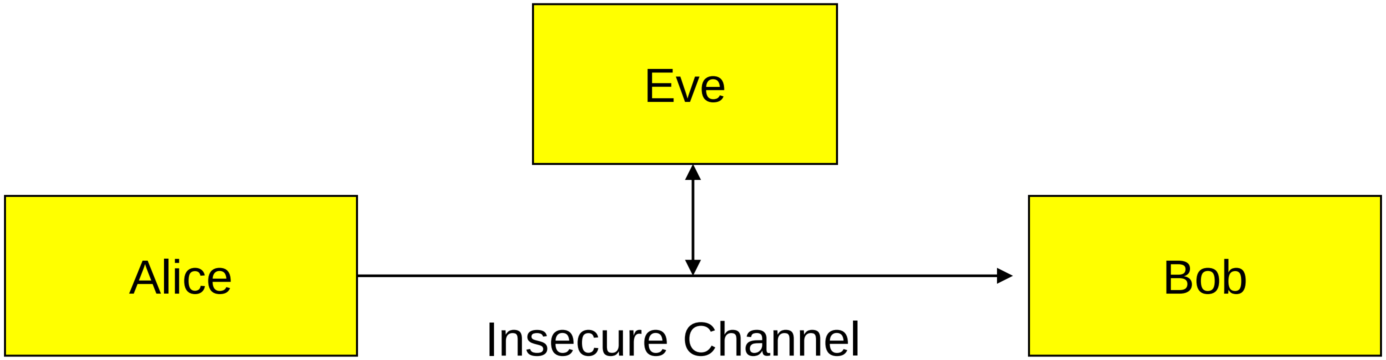
Yuan Hong
Department of Computer Science
Illinois Institute of Technology

Chapter 5
Basic Cryptography

DEFINITIONS

- Cryptography = the science (art) of encryption
- Cryptanalysis = the science (art) of breaking encryption
- Cryptology = cryptography + cryptanalysis

CRYPTOGRAPHY GOALS



- Encryption – Prevent Eve from intercepting message
- Authentication – Prevent Eve from impersonating Alice

STARTING AT THE BEGINNING…

Caesar cipher: a → d, b → e … z → c

- hello → khoor

- "hello" is the ***plaintext***
- "khoor" is the ***ciphertext***

- How can we break (***cryptanalyze***) Caesar cipher?

- Insertion method:

  Find Roman, insert bamboo shoots under fingernails.

FAIL, CAESAR

Why does the Caesar cipher fail so miserably?

- Anyone who knows the design of the cipher can break it. Everyone uses the same cipher!

- **Fundamental rule of modern cryptology**:
- Complete design of cipher should be assumed to be known to adversaries

THE CALIGULA CIPHER

Caesar cipher had a fixed shift of 3.

- Why not let the sender and the receiver agree on what shift to use?

- Shift value is the ***key***

- Design is public, key is kept private

BREAKING THE CALIGULA CIPHER

Given ciphertext "q," any plaintext letter is possible.

- Cannot break Caligula cipher if only given a single message, consisting of a single character!

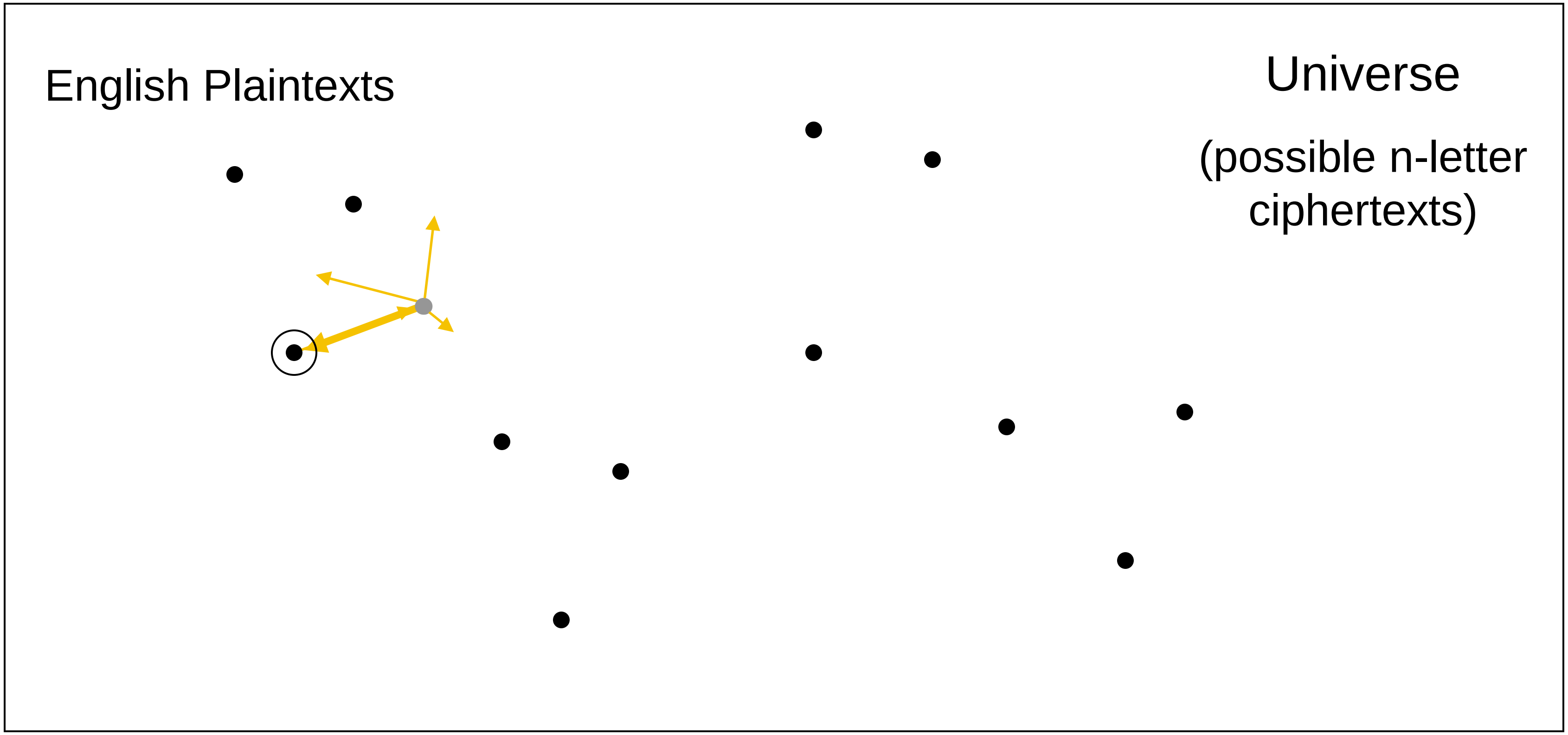
- What about multi-letter English messages?

- Decrypt dbmjhvmb (***known ciphertext***)

- ***Exhaustive Search*:** Try all keys
  - only a shift of 1 gives reasonable plaintext
  - (caligula)

WHEN DOES EXHAUSTIVE SEARCH WORK?

Crypanalysis possible because English messages do not look like random messages.



EXHAUSTIVE SEARCH (CONT'D)

When does this analysis work?

- #(plaintexts) *#(keys) << #(possible ciphertexts)
- 26 * 26 > 26 (Single letter case)
- #(7 letter words) * 26 << 26^7 (7 letter case)

EXHAUSTIVE SEARCH (CONT'D)

Information Theory Approach:

- entropy(plaintext)+entropy(key) < entropy(ciphertext space)

- k characters of English have ≈ 1.1k bits of entropy
- k random characters have ≈ 4.7k bits of entropy
- Key must have 3.6k bits of entropy: $2^{3.6k}$ keys

- ***Unicity Distance*****:** value of k for which space of keys no longer has enough entropy to fight exhaustive search.

In cryptography, **unicity distance** is the length of an original ciphertext needed to break the cipher by reducing the number of possible spurious keys to zero in a brute force attack.

AVOIDING EXHAUSTIVE SEARCH

When can we avoid exhaustive search?

- ***Known Plaintext Attack*:**
  - Suppose we knew that "dbmjhvmb" was an encryption of "caligula"
- ***Chosen Plaintext Attack*:**
  - Get someone to encrypt "horse"
- **Chosen Ciphertext Attack:**
  - Get someone to decrypt "sdfxse"

AVOIDING EXHAUSTIVE SEARCH

***Statistical Attack*:**

Exploit regularities in the English language.

- Most common letter in the English language is “e.”
- Given a long message, suppose that the most common letter is “h.” Shift is probably 3.
- Ciphertext only attack.

CALIGULA STRIKES BACK

Key space of 26 is not enough.

- ***Poly-alphabetic Cipher*****:** Choose a multi-letter keyword: "cat"
    - +supercalifragilistic
    - +catcatcatcatcatcatca
    - ____________________
    - =vvjhswdmcisujjfltnld
- c=shift of 3, a=shift of 1, t=shift of 20

HOW DO OUR ATTACKS FARE?

- **Exhaustive Search:**

  Still possible in principle if plaintext is long

  enough (unicity distance) but much more exhausting

- **Known Plaintext, Chosen Plaintext /Ciphertext:** Works like a charm.

- **Statistical Attack:** Less structure, can still use letter frequencies.

KILLING EXHAUSTIVE SEARCH

Counting/Unicity distance argument implies that for any fixed key size, exhaustive search possible if enough ciphertext is available.

- Solution: Use a key that's **as long as** the plaintext and ciphertext.

```
+supercalifragilistic
+blaireisaslaveofbush
_____________________
=ugqnjhejydbcnaouobk
```

CRYPTOGRAPHIC ATTACKS

- Ciphertext only: attacker has only ciphertext
- Known plaintext: attacker has plaintext and corresponding ciphertext
- Chosen plaintext: attacker can encrypt messages of his choosing
- Distinguishing attack: an attacker can distinguish your cipher from an ideal cipher (random permutation)
- A cipher must be secure against all these attacks

KERCKHOFFS' PRINCIPLE

- The security of an encryption system must depend only on the key, not the secrecy of the algorithm
- Secure systems use published algorithms (PGP, OpenSSL, Truecrypt)

PROVABLE SECURITY

- There is no such thing as a provably secure system
- Proof of unbreakable encryption does not prove the system is secure
- The only provably secure encryption is the one-time pad: $C = P + K$, where K is as long as P and never reused
- Systems are believed secure only when many people try and fail to break them

THE ONE-TIME PAD

All the attacks required that someone use the same key twice.

- So only use it once, dummy!

$$
\begin{array}{r}
01101010100101010100 \\
\oplus\ 10110101001011010101 \\
\hline
11011111101110000001
\end{array}
$$

- Given good random bits, completely secure!

BREAKING THE ONE-TIME PAD

Suppose Martha Stewart wishes to send either "buy" or "sell" to her broker.

- We'd sort of like to know too…

  - +buy
  - +xqr
  - ______
  - =zmq

zmq

zmq

  - +zmq
  - -xqr
  - ______
  - =buy

WHAT IS INDISTINGUISHABILITY?

Allow the adversary to pick any two messages M_0 and M_1.

The encryptor flips a random bit r, and encrypts message M_r, obtaining ciphertext C.

The adversary gets C, tries to guess r.

***Advantage*=Pr[C guesses right]-Pr[C guesses wrong].**

Want advantage to be extremely small.

PRACTICAL ISSUES

One time pad (with other safeguards) is extremely secure.

- One time pads take up a lot of bits
- How do we share a one-time pad? Especially if we never meet.
- Must prevent reuse of the one-time pad.

MAKING DO WITH SMALL KEYS

Exhaustive search is not a viable option when the key size is large enough.

- How large is large enough?
  - 40 bits a pathetic joke
  - 56 bits (DES) doable, though expensive
  - 128 bits seems awfully difficult

- What about the poly-alphabetic cipher?
  - Large key size, still easy to attack
  - Encrypts pieces of message, many tiny keys

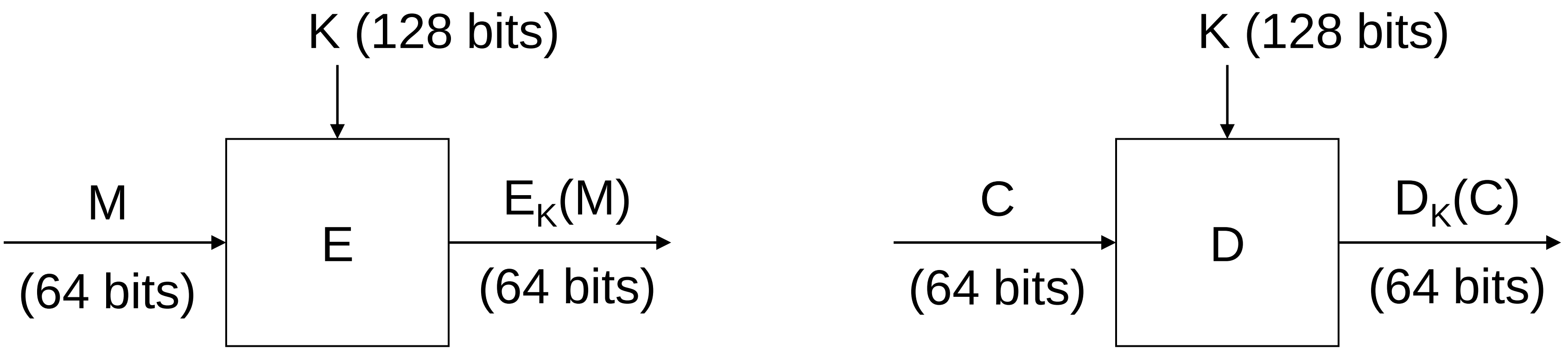
CRYPTOGRAPHIC ALGORITHMS

- Block ciphers (secret/symmetric key)
- Hashes & MAC (keyed hashes)
- Public Key Crypto and Diffie-Hellman key exchange

BLOCK CIPHERS

Staple of cryptography

- Avoid weakness of poly-alphabetic cipher – every bit of output influenced by every bit of input, key.



$$D_K(E_K(M))=M$$

LET'S MAKE A CODE

Start with message: $M=M_1 \cdots M_m$

- $q=M_1 \wedge M_{37}$; $C_5=q \oplus M_8$
- $r=M_3 \oplus (C_5 \wedge M_6)$; $C_1=r \oplus M_2 \oplus M_{63}$
- …
- Output: $C=C_1 \cdots C_m$

- Where's the key?

LET'S TRY HARDER

Start with message: $M=M_1 \cdots M_m$ and $K=K_1 \cdots K_k$

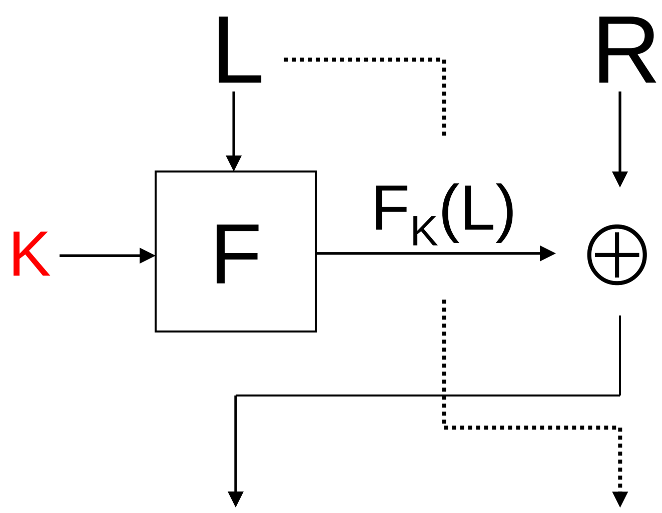
- $q=M_1 \wedge M_{37}$; $C_5 = q \oplus M_8 \oplus K_4$
- $r=M_3 \oplus (C_5 \wedge M_6)$; $C_1 = r \oplus M_2 \oplus (M_{63} \wedge K_2)$
- …
- Output: $C=C_1 \cdots C_m = E_K(M)$

ACHIEVING INVERTIBILITY

Solution 1: **Make sure that the sequence of little steps you took are all reversible.**

Solution 2: **Use one-time pad.**

- $M=LR$



- $E_K(M)=F_K(L)\oplus R$, L

IS THIS INVERTIBLE?

Yes! **Have L for free**

Given L, K, can compute $F_K(L)$

Given $F_K(L) \oplus R$, $F_K(L)$, can compute R

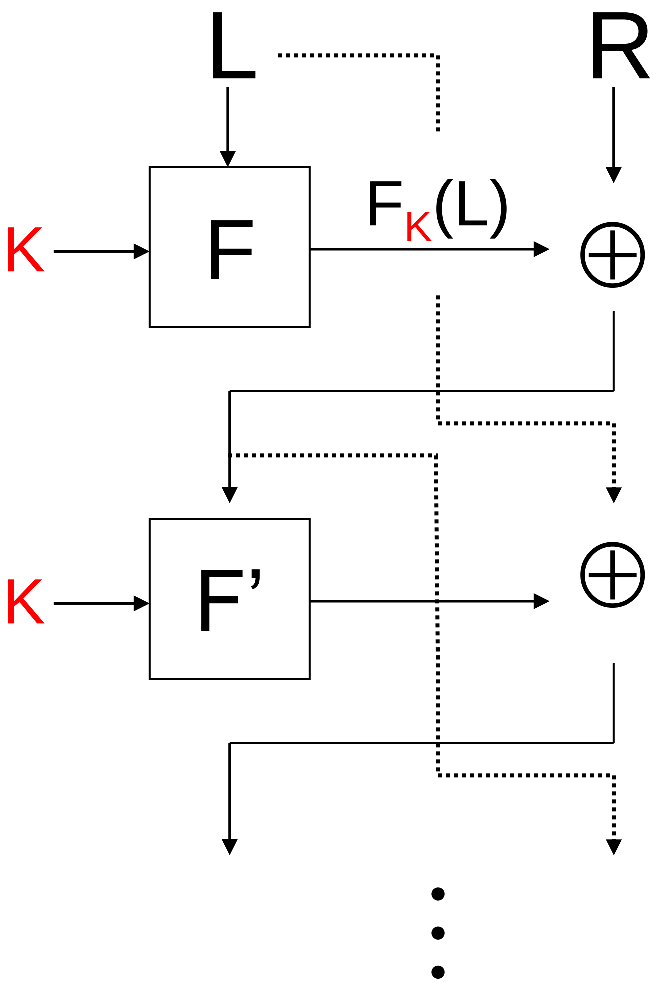
M=LR

- Is this secure?
- No! Eavesdropper learns L for free!
- Solution: Do it again and again…

THE FEISTEL NETWORK

- How many times are needed?

- 1 clearly bad,
- 2 also bad,
- 3 ok if one really trusts Fs
- DES uses 16
- Maybe should have had more…



ARE WE SECURE YET?

- Sequence 1: “buy”, “buy”
- Sequence 2: “buy”, “sell”

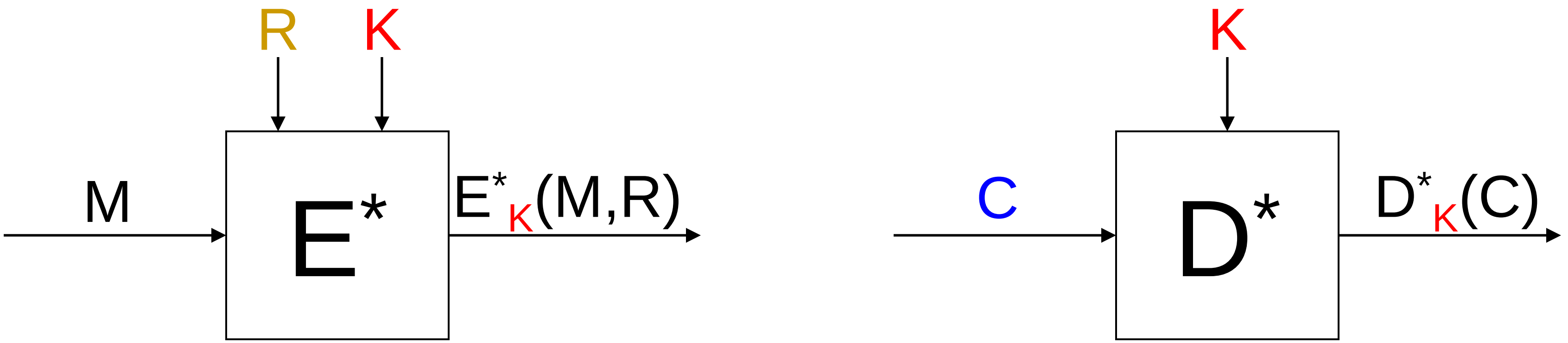
- Martha sends: 010100100, 101110101

- Lesson: Block encryption paradigm is still not enough to obtain security (even against passive eavesdroppers).

- Encryption must have randomized component.
- The same message must look different each time it is encrypted.

PROBABILISTIC ENCRYPTION

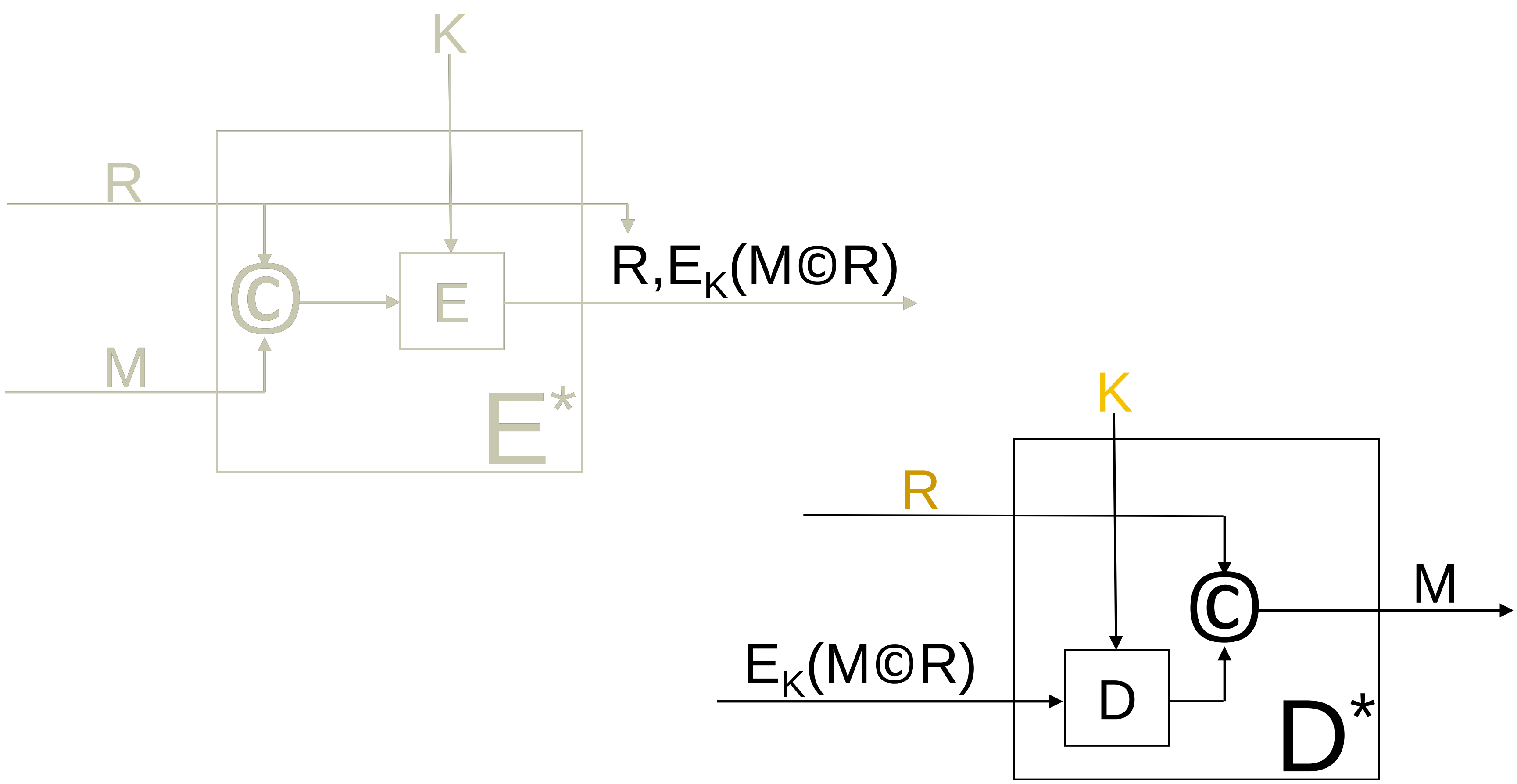
- Solution: Add randomness to the encryption process.



$$(\forall R)\ D^*_K(E^*_K(M,R))=M$$

HOW DO WE ADD RANDOMNESS?

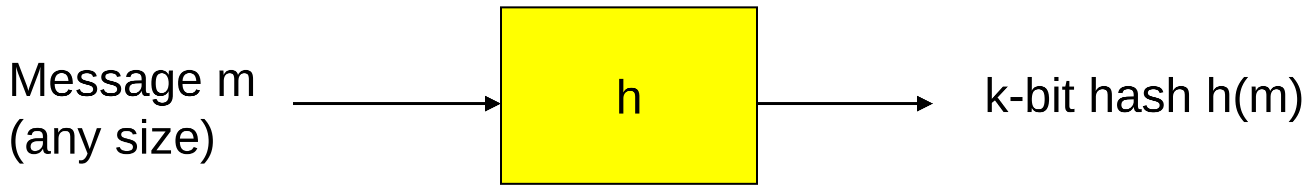
- Use one-time pad trick! (sort of)



CRYPTOGRAPHIC ALGORITHMS

- Block ciphers (secret/symmetric key)
- Hashes & MAC (keyed hashes)
- Public Key Crypto and Diffie-Hellman key exchange

SECURE HASH FUNCTIONS



- Goals
  - **Collision resistance**: it takes $2^{k/2}$ work to find any m_1, m_2 such that $h(m_1) = h(m_2)$.
  - **First preimage resistance**: given h(m), it takes 2^k work to find m.
  - **Second preimage resistance**: given m_1, it takes 2^k work to find m_2 such that $h(m_1) = h(m_2)$.

WHAT IS A HASH FUNCTION ANYWAY?

A *cryptographic hash function* H maps n bit numbers to k bit numbers. (k is a security parameter)

H should be “completely unstructured.”

What does that mean?

A. Looks like a random function.

B. Hard to find collisions

COLLISION INTRACTIBILITY

Suppose $n>k$. Then by a counting argument, there exist lots of n-bit strings $x_1,\ldots,x_m$, that hash to the same value $y(m¸2^{n-k})$.

But can we find two strings x_1 and x_2 that collide?

Brute force: Try $\approx 2^{k/2}$ and you should find a collision.

Collision Intractible Hash Functions: Can't find a collision.

HASH APPLICATIONS

- Faster digital signatures: Alice signs h(P) instead of P.
- Password verification (e.g., UNIX) without storing passwords.
- Strong pseudo-random number generation.
- Message Authentication Code (MAC).

HASH EXAMPLES

- MD2, MD4, MD5 – 128 bits (broken,
  http://eprint.iacr.org/2004/199.pdf
  http://eprint.iacr.org/2006/105.pdf)
- SHA-1 – 160 bits
- SHA-256, 384, 512 bits
  http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf
- Whirlpool – 512 bits
- Tiger – 192 bits
- Many proposed hashes have been broken.
  http://paginas.terra.com.br/informatica/paulobarreto/hflounge.html

MESSAGE AUTHENTICATION CODE (MAC)

HMAC(K, m) =
h(K xor 0x5c5c…|| h(K xor 0x3c3c… || m))

- h = SHA-1 or MD5
- K = key
- m = message

Can only be computed if you know K.

FIPS Pub 198

CRYPTOGRAPHIC ALGORITHMS

- Block ciphers (secret/symmetric key)
- Hashes & MAC (keyed hashes)
- Public Key Crypto and Diffie-Hellman key exchange

PUBLIC KEY CRYPTOGRAPHY

Two keys

- *Private key* known only to individual
- *Public key* available to anyone
  - Public key, private key inverses

Idea

- **Confidentiality**: encipher using public key, decipher using private key

- **Integrity/authentication**: encipher using private key, decipher using public one

REQUIREMENTS

1. It must be computationally easy to encipher or decipher a message given the appropriate key

2. It must be computationally infeasible to derive the private key from the public key

3. It must be computationally infeasible to determine the private key from a chosen plaintext attack

DIFFIE-HELLMAN

Compute a common, shared key

- Called a *symmetric key exchange protocol*

Based on discrete logarithm problem

- Given integers n and g and prime number p, compute k such that $n = g^k \bmod p$
- Solutions known for small p
- Solutions computationally infeasible as p grows large

ALGORITHM

Constants: prime p, integer $g \neq 0, 1, p-1$

- Known to all participants

Alice chooses private key *kAlice*, computes public key *KAlice* = g^{kAlice} mod p. *So does Bob*

To communicate with Bob, Alice computes *Kshared* = $KBob^{kAlice}$ mod p

To communicate with Alice, Bob computes *Kshared* = $KAlice^{kBob}$ mod p

- It can be shown these keys are equal

EXAMPLE

Assume p = 53 and g = 17

Alice chooses *kAlice* = 5

- Then *KAlice* = 17^5 mod 53 = 40

Bob chooses *kBob* = 7

- Then *KBob* = 17^7 mod 53 = 6

Shared key:

- $KBob^{kAlice}$ mod p = 6^5 mod 53 = 38
- $KAlice^{kBob}$ mod p = 40^7 mod 53 = 38

RSA

Exponentiation cipher

Relies on the difficulty of determining the <u>number</u> of numbers relatively prime to a large integer *n*

BACKGROUND

Totient function $\phi(n)$

- **Number** of positive integers less than *n* and relatively prime to *n*
- *Relatively prime* means with no factors in common with *n*

Example: $\phi(10) = 4$

- 1, 3, 7, 9 are relatively prime to 10

Example: $\phi(21) = 12$

- 1, 2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20 are relatively prime to 21

ALGORITHM

Choose two large prime numbers p, q

- Let $n = pq$; then $\phi(n) = (p-1)(q-1)$
- Choose $e < n$ such that e is relatively prime to $\phi(n)$.
- Compute d such that $\boldsymbol{ed \bmod \phi(n) = 1}$

Public key: (e, n); Private key: (d, n)

Encipher: $c = m^e \bmod n$

Decipher: $m = c^d \bmod n$

EXAMPLE: CONFIDENTIALITY

Take p = 7, q = 11, so n = 77 and $\phi(n)$ = 60

Alice chooses $e = 17$, making $d = 53$

Bob wants to send Alice secret message HELLO (07 04 11 11 14)

- $07^{17} \bmod 77 = 28$
- $04^{17} \bmod 77 = 16$
- $11^{17} \bmod 77 = 44$
- $11^{17} \bmod 77 = 44$
- $14^{17} \bmod 77 = 42$

Bob sends 28 16 44 44 42

EXAMPLE

Alice receives 28 16 44 44 42

Alice uses private key, *d* = 53, to decrypt message:

- 28^{53} mod 77 = 07
- 16^{53} mod 77 = 04
- 44^{53} mod 77 = 11
- 44^{53} mod 77 = 11
- 42^{53} mod 77 = 14

Alice translates message to letters to read HELLO

- No one else could read it, as only Alice knows her private key and that is needed for decryption

EXAMPLE: INTEGRITY/AUTHENTICATION

Take p = 7, q = 11, so n = 77 and $\phi(n)$ = 60

Alice chooses e = 17, making d = 53

Alice wants to send Bob message HELLO (07 04 11 11 14) so Bob knows it is what Alice sent (no changes in transit, and authenticated)

- $07^{53} \bmod 77 = 35$
- $04^{53} \bmod 77 = 09$
- $11^{53} \bmod 77 = 44$
- $11^{53} \bmod 77 = 44$
- $14^{53} \bmod 77 = 49$

Alice sends 35 09 44 44 49

EXAMPLE

Bob receives 35 09 44 44 49

Bob uses Alice's public key, *e* = 17, *n* = 77, to decrypt message:

- $35^{17} \bmod 77 = 07$
- $09^{17} \bmod 77 = 04$
- $44^{17} \bmod 77 = 11$
- $44^{17} \bmod 77 = 11$
- $49^{17} \bmod 77 = 14$

Bob translates message to letters to read HELLO

- Alice sent it as only she knows her private key, so no one else could have enciphered it
- If (enciphered) message's blocks (letters) altered in transit, would not decrypt properly

EXAMPLE: BOTH

Alice wants to send Bob message HELLO both enciphered and authenticated (integrity-checked)

- Alice's keys: public (17, 77); private (53, 77)
- Bob's keys: public (37, 77); private (13, 77)

Alice enciphers HELLO (07 04 11 11 14):

- $(07^{53} \bmod 77)^{37} \bmod 77 = 07$
- $(04^{53} \bmod 77)^{37} \bmod 77 = 37$
- $(11^{53} \bmod 77)^{37} \bmod 77 = 44$
- $(11^{53} \bmod 77)^{37} \bmod 77 = 44$
- $(14^{53} \bmod 77)^{37} \bmod 77 = 14$

Alice sends 07 37 44 44 14

SECURITY SERVICES

Confidentiality

- Only the owner of the private key knows it, so text enciphered with public key cannot be read by anyone except the owner of the private key

Authentication

- Only the owner of the private key knows it, so text enciphered with private key must have been generated by the owner

MORE SECURITY SERVICES

Integrity

- Enciphered letters cannot be changed undetectably without knowing private key

Non-Repudiation

- Message enciphered with private key came from someone who knew it

COMMUTATIVE ENCRYPTION

We saw that Alice encrypted with her private key, and then with Bob’s public key

Could use both public keys (or both private keys)

Indeed,

- Since both encryption and decryption involves exponentiation, order does not matter

Commutative

FINDING IF TWO NUMBERS ARE EQUAL

Alice has a number a

Bob has a number b

Alice creates a key-pair PK_a, SK_a

Bob creates a key-pair PK_b, SK_b

Alice encrypts a with PK_a

Bob encrypts b with PK_b

Now they exchange and encrypt again with their keys

If $PK_a\,(PK_b\,(a)) = PK_b\,(PK_a\,(b))$, then $a = b$

ACKNOWLEDGMENTS

ILLINOIS INSTITUTE OF TECHNOLOGY

Note: Some of the slides in this lecture are based on material created by

- Dr. Joe Kilian at IDA Center for Communications Research
- Dr. Jiangtao Li at Google
- Dr. Jaideep Vaidya at Rutgers