CMPUT 626 - A2
Machine Learning and Practical Privacy

Thinking About Cryptography 1

Fall 2023, Tuesday/Thursday 3:30-4:50pm

Lecture Section Update

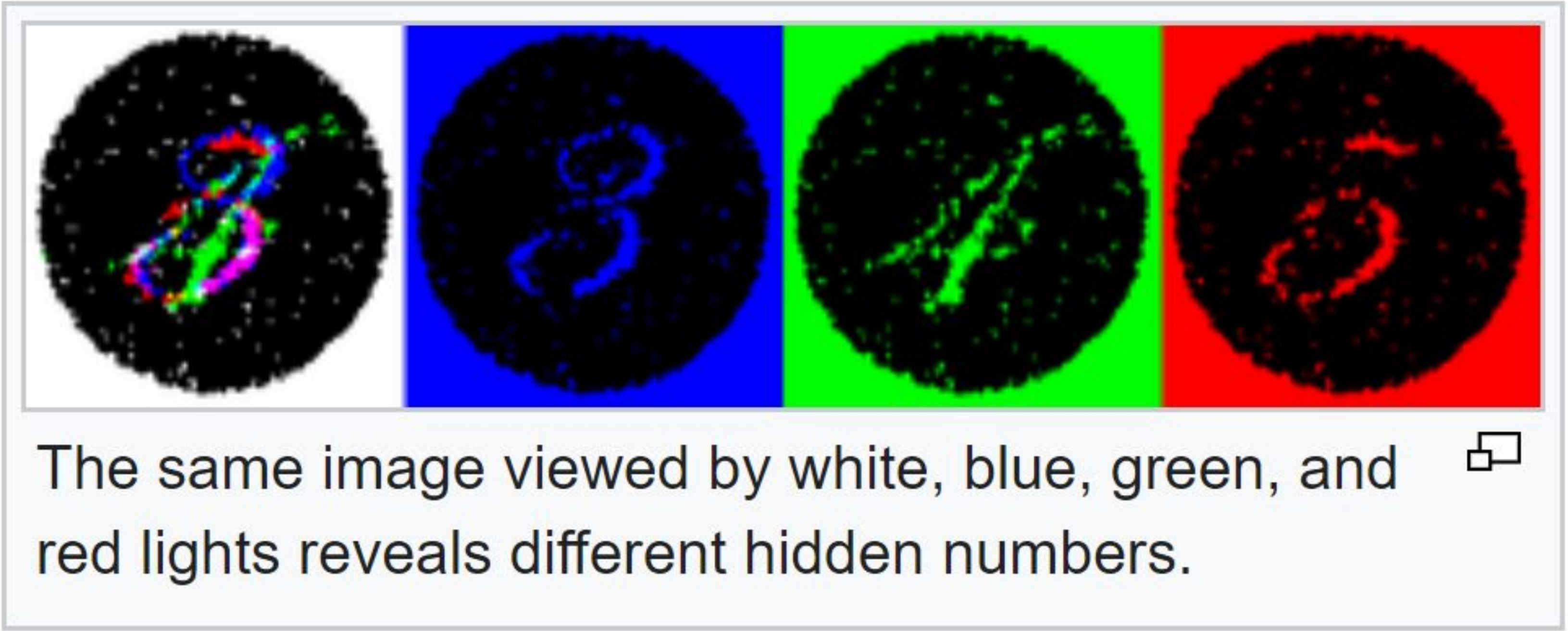
Week	Tuesday	Thursay
One: September 5th and 7th	Course overview, Privacy Part 1 Bailey	Cryptography Part 1 Bailey
Two: September 12th and 14th	Cryptography Part 2 Bailey	Ethics, law, and policy Bailey
Three: September 19th and 21st	Privacy Part 2 Bailey	Privacy Part 3 Bailey

Learning Outcomes

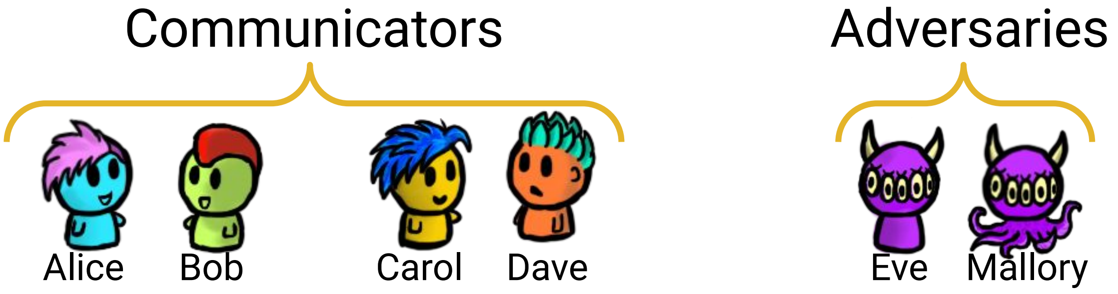
- Identify attack techniques and apply them (cryptanalysis)
- Explain building blocks of modern cryptography
- Explain how modern cryptography properties arose

Goal: Basically, know what cryptography tools exist and how to securely use them. <u>Build a foundation of primitives</u> for more complicated “applied cryptography” later.

Steganography- Secretly “hidden” messages

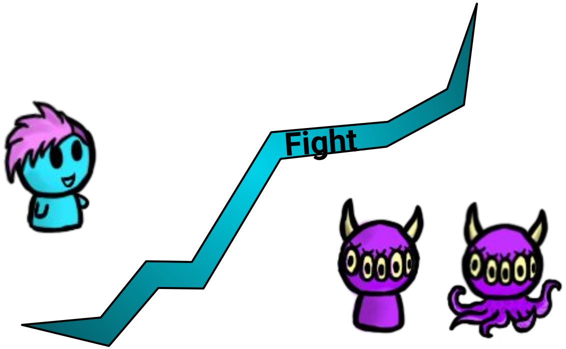


Cryptography - Writing “secret” messages

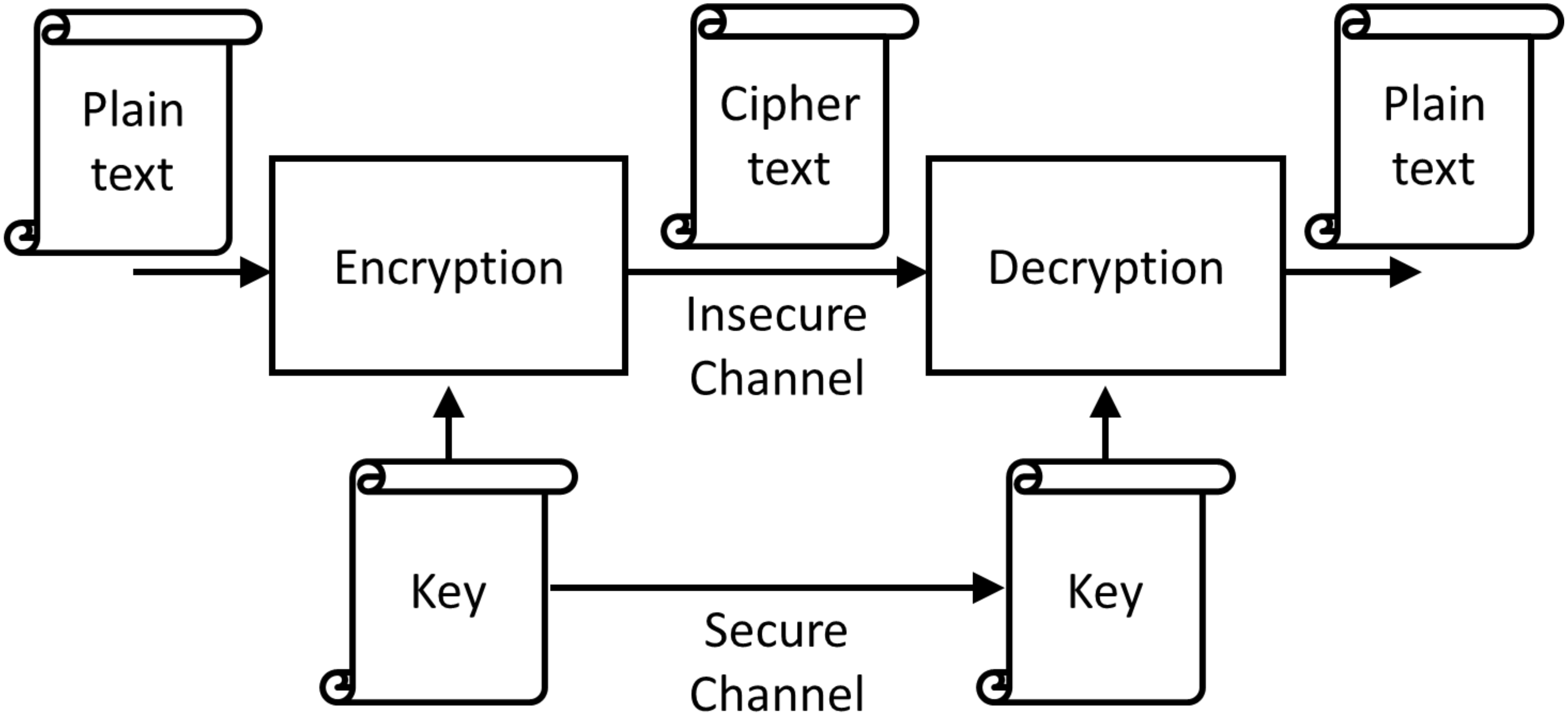


CIA and Cryptography

- Confidentiality, prevent Eve **reading** Alice’s messages
- Integrity, prevent Mallory from **changing** Alice’s messages
- Authenticity, Prevent Mallory from **impersonating** Alice



Cryptography - Path for Secret Messages



Historical Ciphers: Example One

FUBSWRJUDSKB

CRYPTOGRAPHY

Caesar Cipher

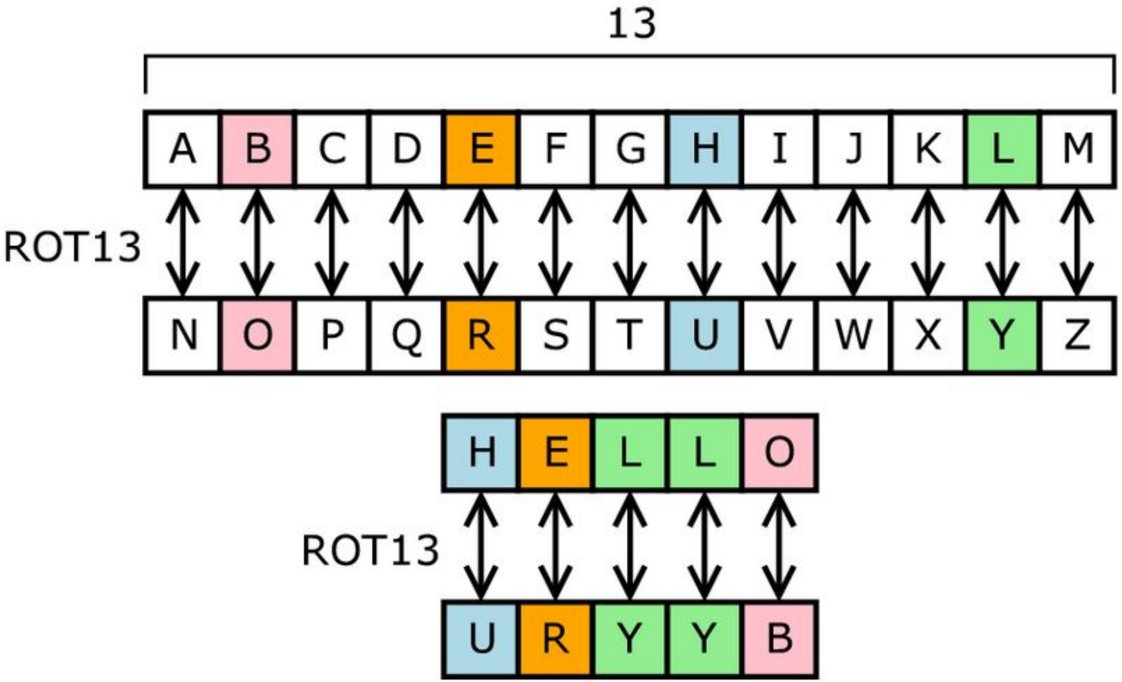


Image source: wikipedia

Cryptanalysis - Analyzing “secret” messages

Mwahaha

We will learn the secretsssss.

Historical Ciphers: Example Two

gsrh xlfihv rh zylfg xibkgltizksb uli gsv urihg gsivv dvvph. zmw gsvm zkkorvw xibkgltizksb uli kirezxb zmw hvxfirgb lu wzgz.

English Frequency

A	11.7%	
B	4.4%	
C	5.2%	
D	3.2%	
E	2.8%	
F	4%	
G	1.6%	
H	4.2%	
I	7.3%	
J	0.51%	
K	0.86%	
L	2.4%	
M	3.8%	

N	2.3%	
O	7.6%	
P	4.3%	
Q	0.22%	
R	2.8%	
S	6.7%	
T	16%	
U	1.2%	
V	0.82%	
W	5.5%	
X	0.045%	
Y	0.76%	
Z	0.045%	

Historical Ciphers: Example Two

gsrh xlfihv rh zylfg xibkgltizksb uli **gs**v urihg **gs**ivv dvvph. zmw **gs**vm zkkorvw xibkgltizksb uli kirezxb zmw hvxfirgb lu wzgz.

Historical Ciphers: Example Two

gsrh xlfihv rh zylfg xibkgltizksb uli **gs**v urihg **gs**ivv dvvph. zmw **gs**vm zkkorvw xibkgltizksb uli kirezxb zmw hvxfirgb lu wzgz.

This course is about cryptography for **th**e first **th**ree weeks. And **th**en applied cryptography for privacy and security of data.

Kerckhoff Principle

The security of a cryptosystem should solely depend on the secrecy of the key, but never on the secrecy of the algorithms.

Historical Ciphers: Example Three

LECTURE SECURITY AND CRYPTOGRAPHY I

↓

LENGECDRCUCATRRPUIYHRTPYEYTISAO

Historical Ciphers: Example Three

LECTURES

ECURITYA

NDCRYPTO

GRAPHYI

↓

LENGECDRCUCATRRPUIYHRTPYEYTISAO

Historical Ciphers: Example Three

Shannon's maxim!!!! (design assuming they'll learn the algorithm

LENGECDRCUCATRRPUIYHRTPYEYTISAO

Shannon’s Maxim and Kerkhoff’s Principle Mean:

- Security shouldn’t rely on the secrecy of the method
- Do use <u>public</u> algorithms with <u>secret</u> “keys”
- The adversaries target…is the key

Key: Easier to change a “short” key than your whole system.
(e.g., Recovery)

Unconditionally Secure: One-Time Pad

Message: $x_0 \; x_1 \; x_2 \; \dots \; x_n$

$\oplus$

Key: $k_0 \; k_1 \; k_2 \; \dots \; k_n$

$=$

Ciphertext: $y_0 \; y_1 \; y_2 \; \dots \; y_n$

Rule: $y_i = x_i + k_i \pmod{2}$

Provably Security for One-Time Pad

<Ciphertext is uniformly distributed independent of the plaintext distribution>

$x_i = 0$ with probability p ($x_i = 1$: 1-p),

$k_i = 0$ with probability 0.5 ($k_i = 1$: 0.5), $y_i = 0$ with probability:

$$
\begin{aligned}
p(y_i = 0) &= p(x_i = 0)\, p(k_i = 0) + p(x_i = 1)\, p(k_i = 1) \\
&= 0.5p + 0.5(1-p) \\
&= 0.5
\end{aligned}
$$

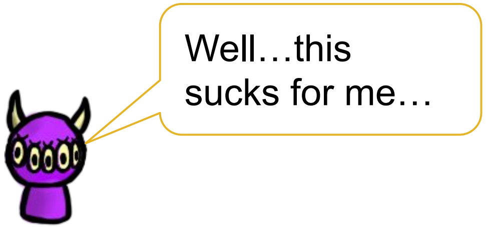
Provably Secure Con’t

Every ciphertext y can be decrypted **into every arbitrary plaintext** x using the key

$$\mathbf{k = yx}$$

Consequently the ciphertext cannot contain any information about the plaintext

Encryption is “deniable”



Act.

What if it is a many-time pad?

Key: K

$\text{Ciphertext}_1 = \text{message}_1$ xor K = 2c1549100043130b1000290a1b

$\text{Ciphertext}_2 = \text{message}_2$ xor K = 3f16421617175203114c020b1c

Your turn, goal: Learn the ciphertexts.

Many-time pad? Messages Lack True Randomness

C_1

C_2

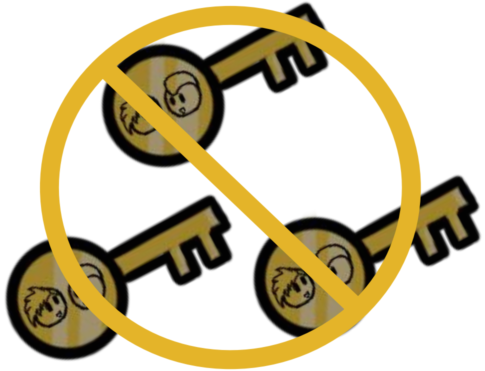
$C_1 \oplus C_2$

M_2

M_1

One-Time Pad - Conditions...

- Key as long as the message
- Key uniformly random
- Only used once



So…Cryptography?

- Simple substitution/transposition is computationally insecure
- One-Time Pad is inefficient over the secure channel

Goal: Securely communicate “a lot” of information on an insecure channel while requiring “limited” communication over a secure channel

Recap: A, B, C versus A and B and C

Substitution is insecure…

Transposition is insecure…

Key reuse using XOR (one-time pad) is insecure…

BUT

Repeat it often enough and it can be widely regarded as secure

Recap: A, B, C versus A and B and C

Substitution is insecure…

Transposition is in…

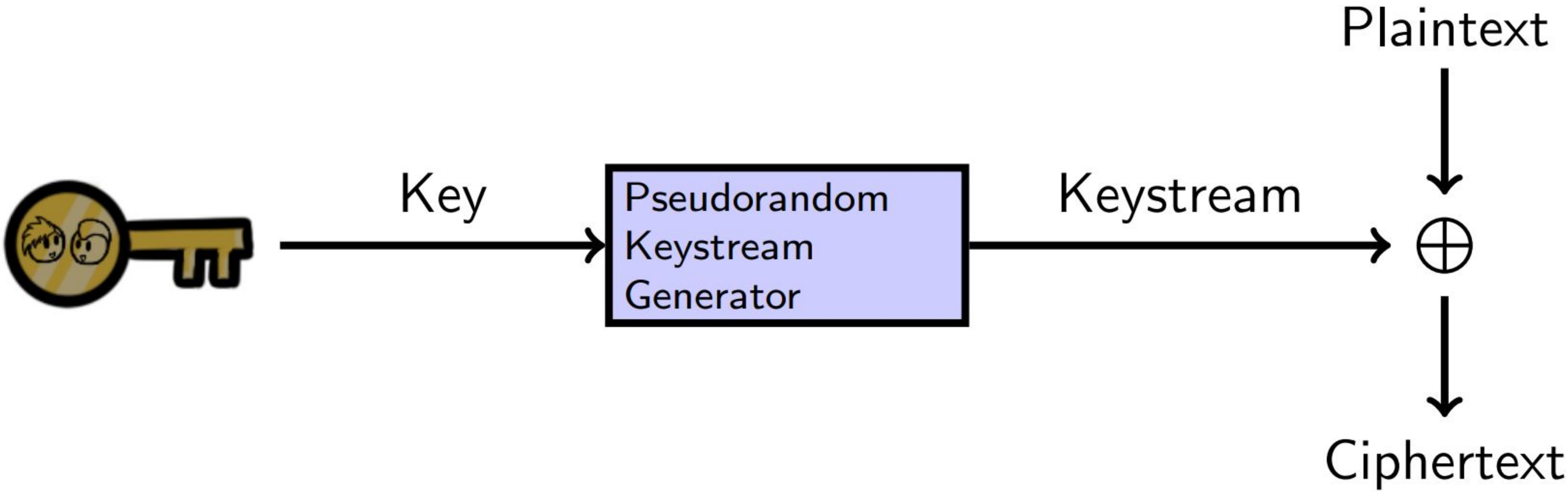
Key reuse u[…]d) is insecure…

BUT

Repeat it often enough and it can be widely regarded as secure

Stream Ciphers and Block Ciphers

Stream Cipher?

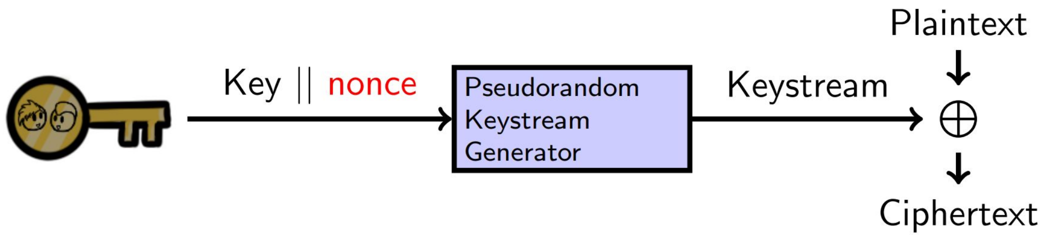


Fun(?) Facts:

- RC4 was the most common stream cipher on the Internet but deprecated.
- ChaCha increasingly popular (Chrome and Android), and SNOW3G in mobile phone networks.

Stream Ciphers Share Conditions with OTP

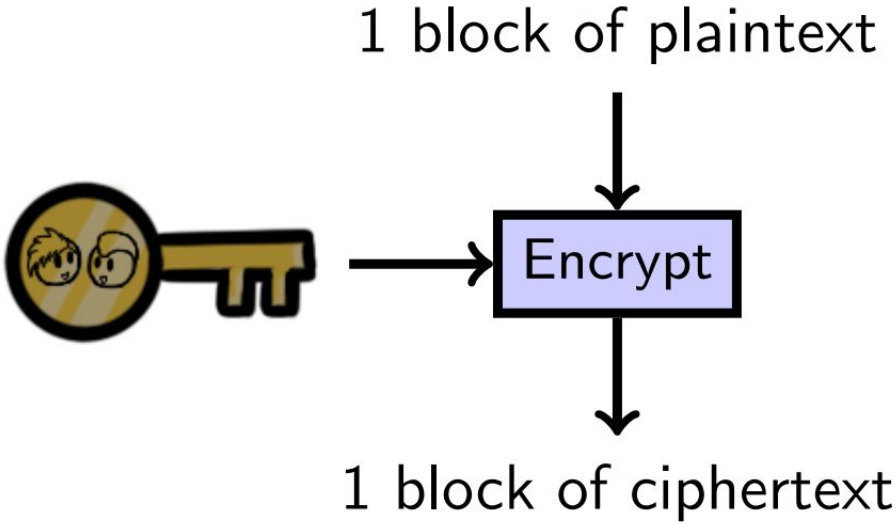
- Stream ciphers can be very fast
  - This is useful if you need to send a lot of data securely
- But they can be tricky to use correctly!
  - We saw the issues of re-using a key! (two-time pad)
  - Solution: concatenate key with nonce (we'll see more about nonces later)



Fun(?) Facts:

- WEP, PPTP are great examples of how not to use stream ciphers

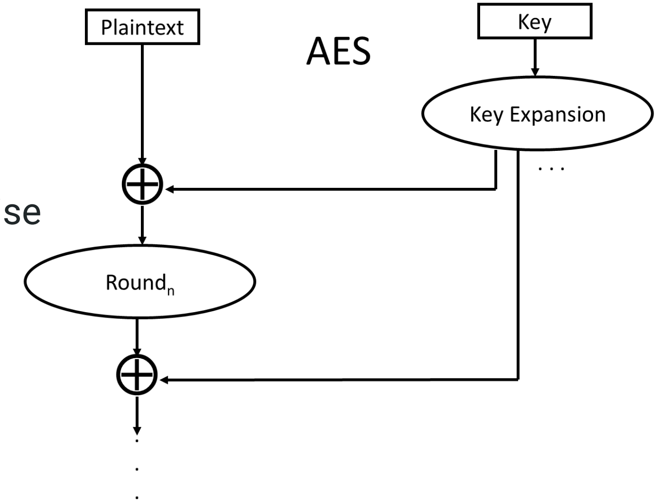
Bit by bit....do you have to?



Block ciphers!!!

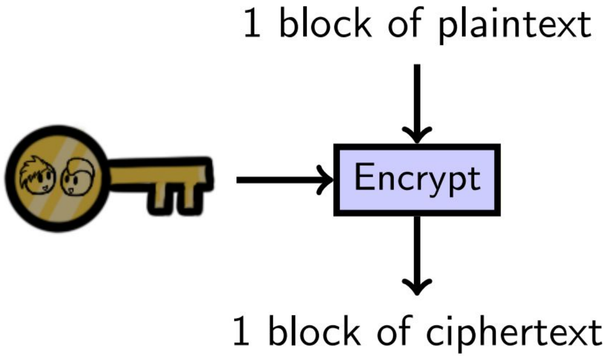
Block Ciphers

- Weakness of streams…one bit at a time?
  - What happens in a stream cipher if you change just one bit of the plaintext?
- Welcome, use of block ciphers
  - Block ciphers operate on the message one block at a time
  - Blocks are usually 64 or 128 bits long
- **AES**, the current standard
  - You better have a very…very good reason to choose otherwise

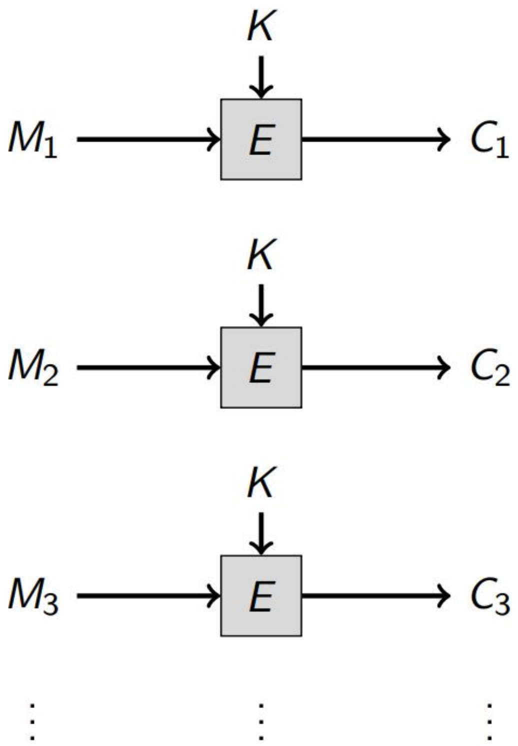


Two Catches with Block Ciphers

- Message is shorter than one block
  - padding
- Message is longer than a block
  - Modes of operation <new concept>

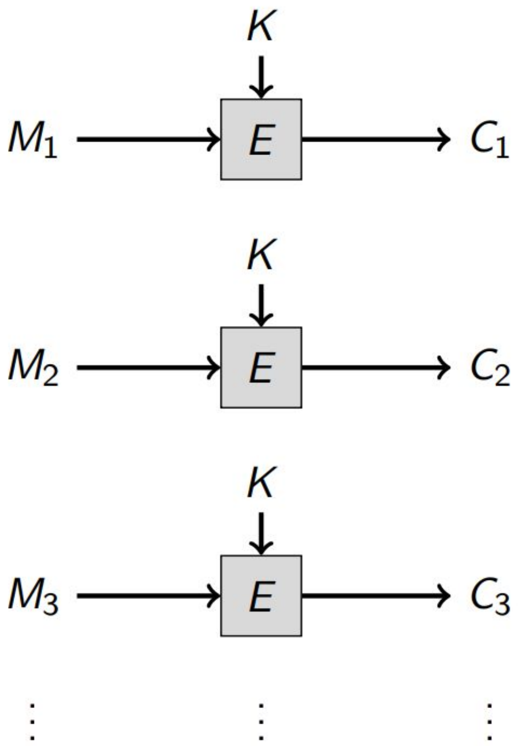


Block Ciphers and Modes of Operation: ECB Mode



- ECB: Electronic Code Book
- Encrypts each successive block separately

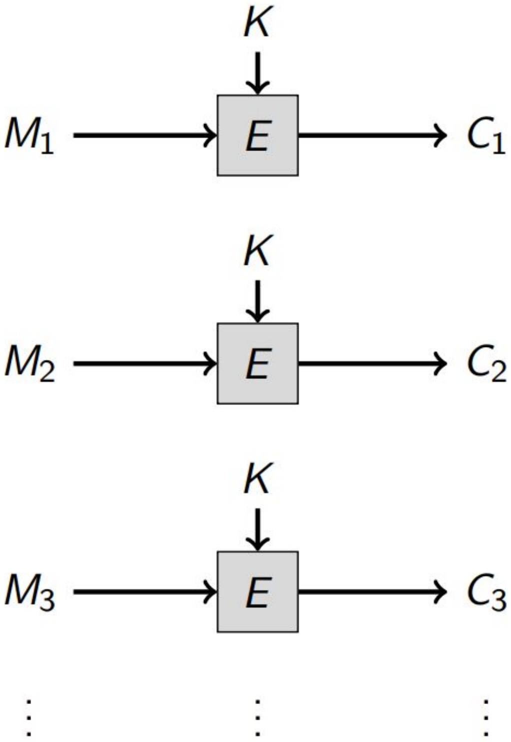
Block Ciphers and Modes of Operation: ECB Mode



- ECB: Electronic Code Book
- Encrypts each successive block separately

Q: What happens if the plaintext M has some blocks that are identical, $M_i = M_j$?

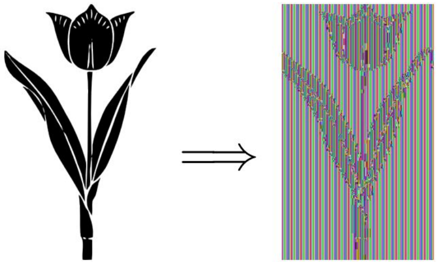
Block Ciphers and Modes of Operation: ECB Mode



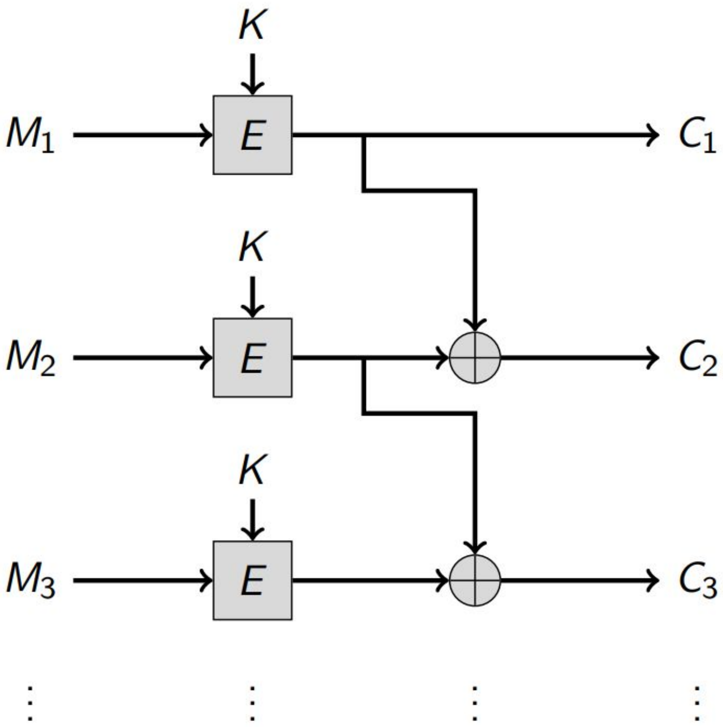
- ECB: Electronic Code Book
- Encrypts each successive block separately

Q: What happens if the plaintext M has some blocks that are identical, $M_i = M_j$?

A: $C_i = E_K(M_i)$, $C_j = E_K(M_j) \Rightarrow C_i = C_j$



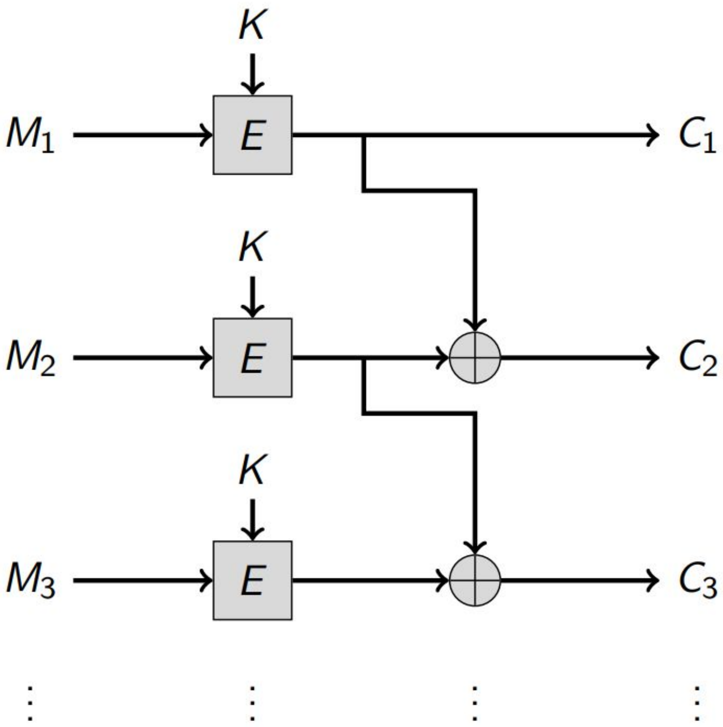
Attempt 1: Fixing ECB$_1$



- Provide “feedback” among different blocks, to avoid repeating patterns…

Q: Fix repeating patterns? Are there other issues?

Attempt 1: Fixing ECB[1]

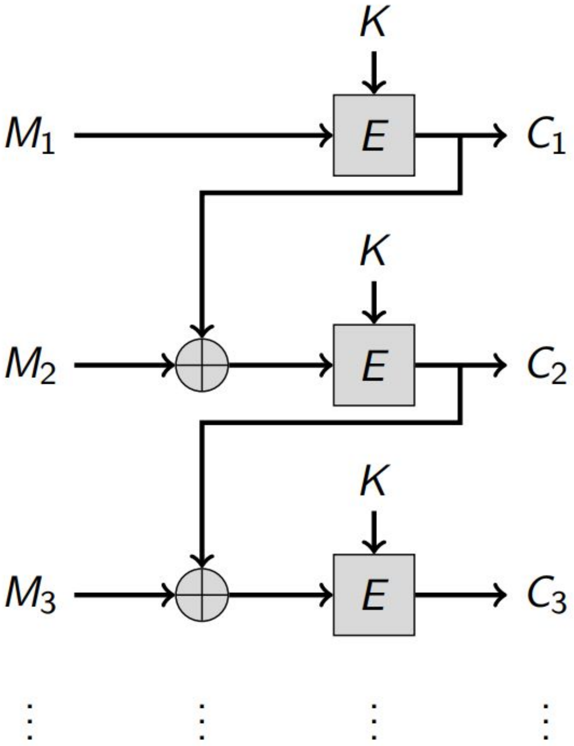


- Provide "feedback" among different blocks, to avoid repeating patterns…

Q: Fix repeating patterns? Are there other issues?

A: We can un-do the XOR <u>if we get all the ciphertexts</u>. This basically does not improve compared to ECB.

Attempt 2: ECB_2!!!

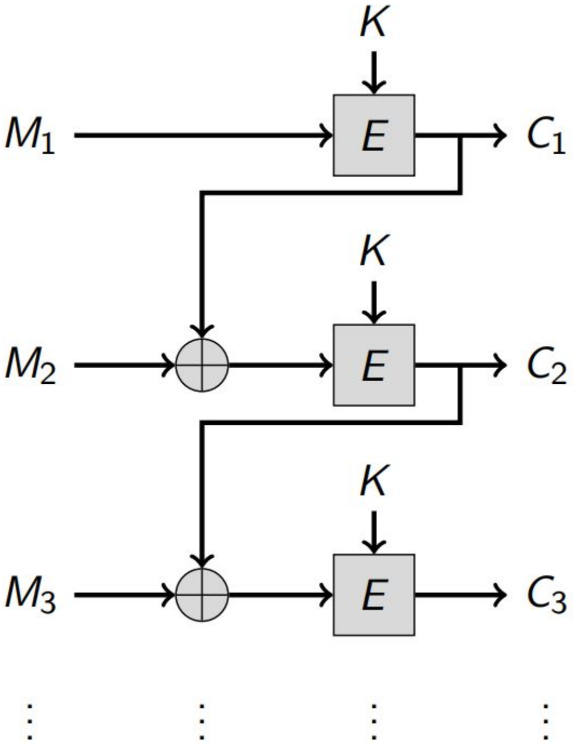


Q: Spot the difference?

Q: Is it fixed this time?

Q: Does this avoid repeating patterns among blocks?

Attempt 2: ECB_2!!!



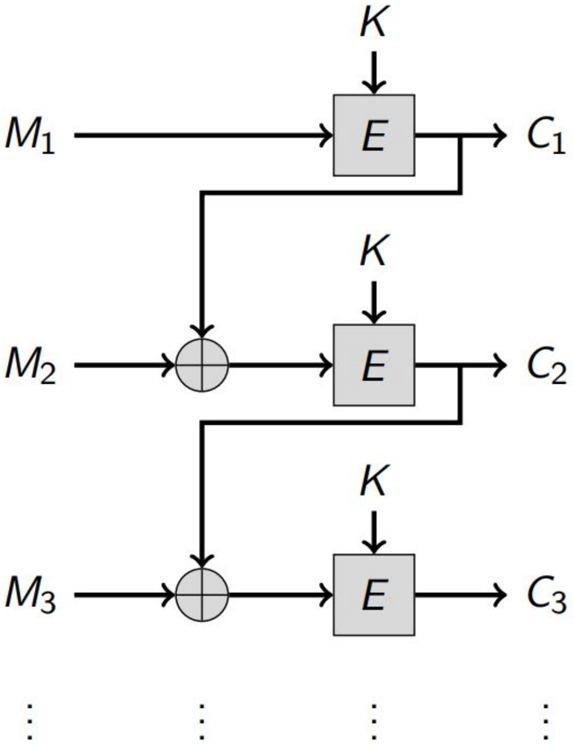
Q: Spot the difference?

Q: Is it fixed this time?

Q: Does this avoid repeating patterns among blocks?

Q: What would happen if we encrypt the message twice with the same key?

Attempt 2: ECB_2!!!



Q: Spot the difference?

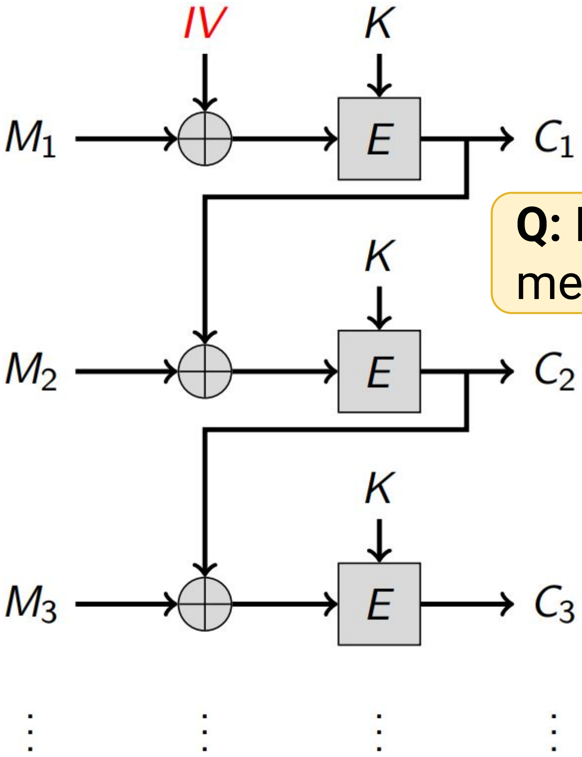
Q: Is it fixed this time?

Q: Does this avoid repeating patterns among blocks?

Q: What would happen if we encrypt the **message twice** with the **same key**?

A: $C_1 = E_K(M), C_2 = E_K(M) \Rightarrow C_1 = C_2$

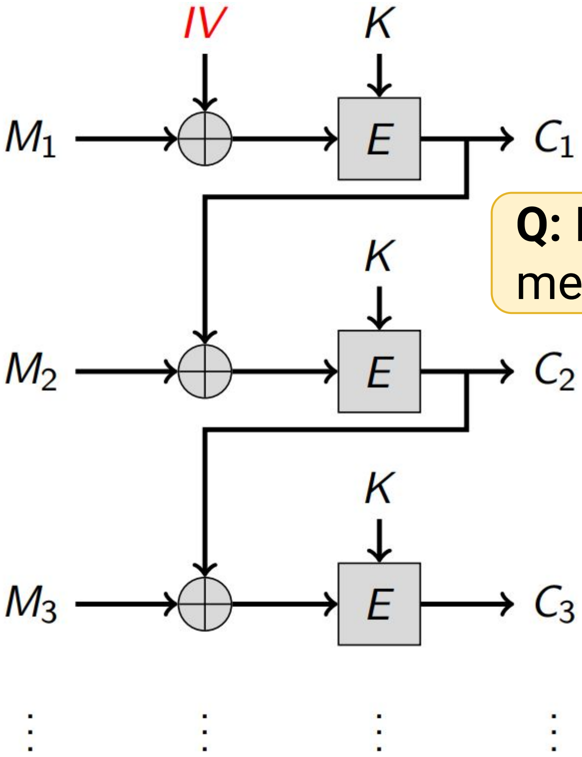
New Plan: CBC Mode



Q: Does this solve the issue of encrypting equal blocks?

Q: Does this solve the issue of encrypting equal messages/plaintexts?

New Plan: CBC Mode

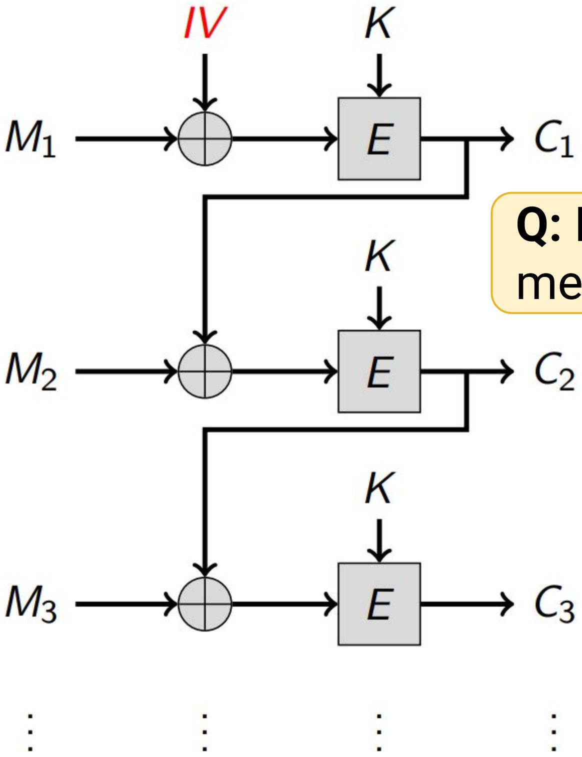


Q: Does this solve the issue of encrypting equal blocks?

Q: Does this solve the issue of encrypting equal messages/plaintexts?

A: Yes!!!

New Plan: CBC Mode



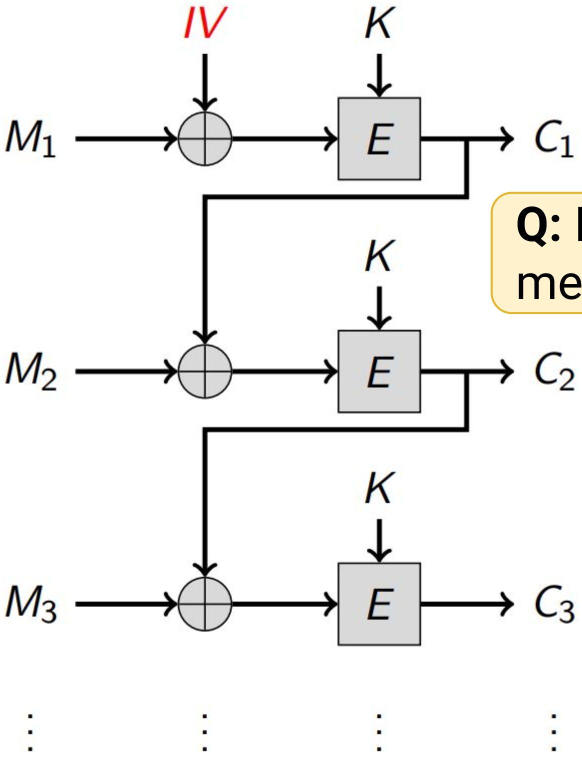
Q: Does this solve the issue of encrypting equal blocks?

Q: Does this solve the issue of encrypting equal messages/plaintexts?

A: Yes!!!

Q: Can we share IV in the clear?

New Plan: CBC Mode



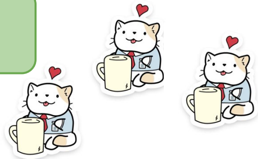
Q: Does this solve the issue of encrypting equal blocks?

Q: Does this solve the issue of encrypting equal messages/plaintexts?

A: Yes!!!

Q: Can we share IV in the clear?

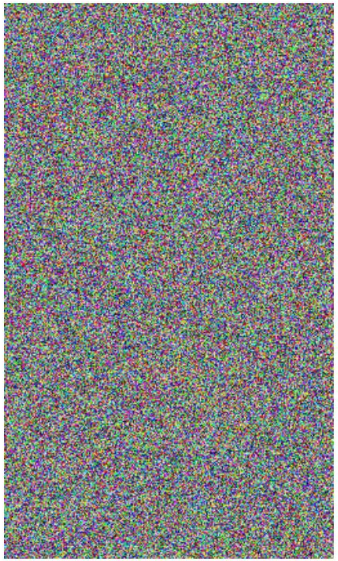
A: Yes!!!



IV, an initialization vector, nonce, salt.

Modes of Operation Collection

- Cipher Block Chaining **(CBC)**, Counter **(CTR)**, and Galois Counter **(GCM)** modes
- Patterns in the plaintext are no longer exposed because these modes involve some kind of “feedback” among different blocks.
- But you need an **IV**



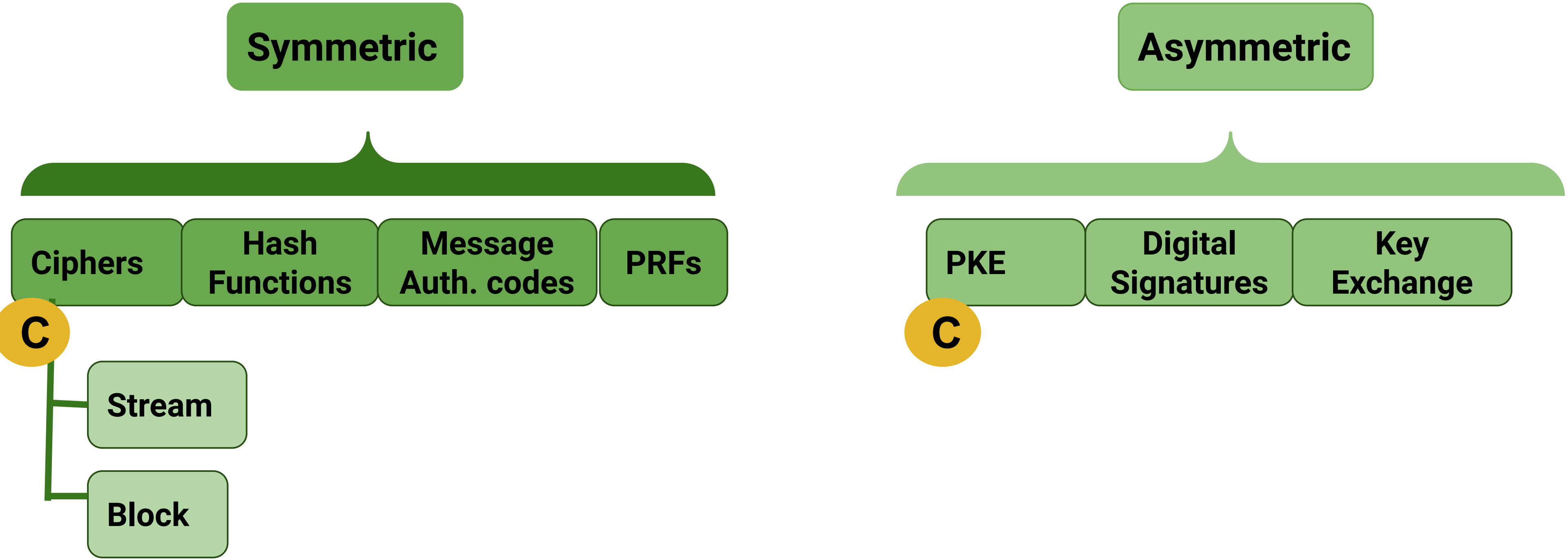
So...now what?

- How do Alice and Bob share the secret key?
    - Meet in person; diplomatic courier...
- In general this is very hard

Or, we invent new technology!!

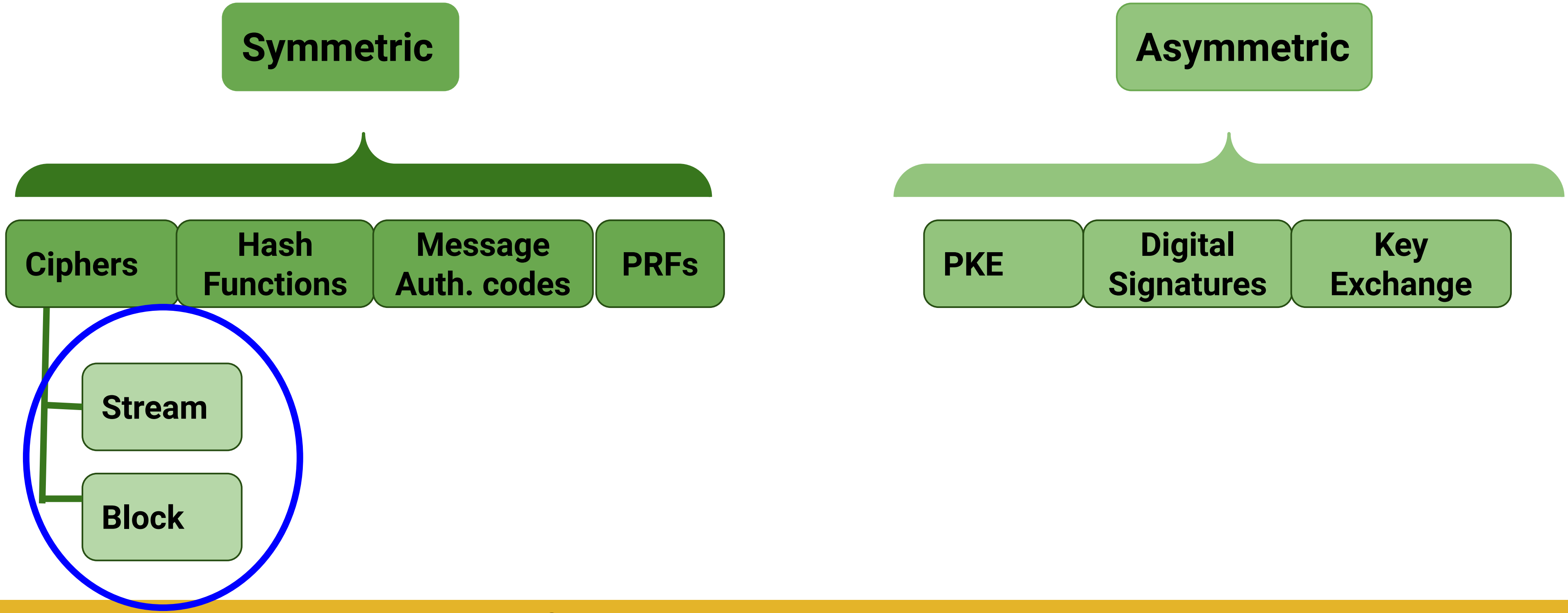
Spoiler Alert: it's already been invented...

Cryptography Organization

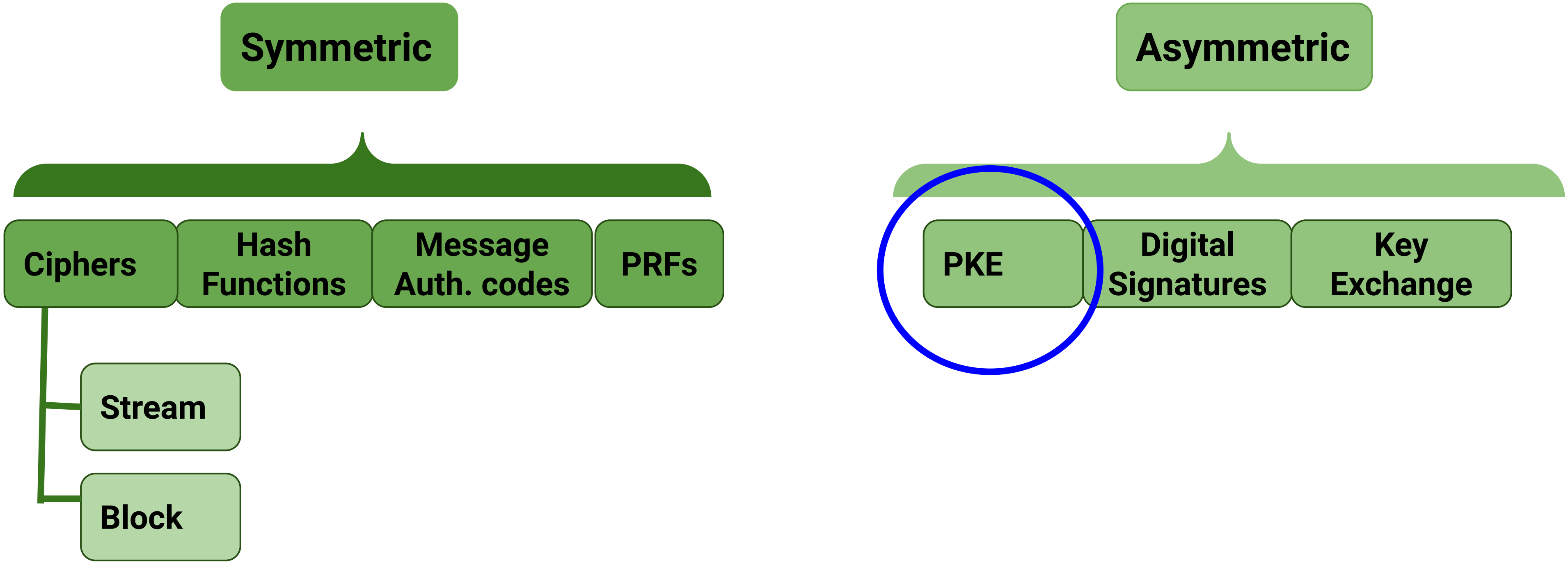


Organization display source: Doug Stebila

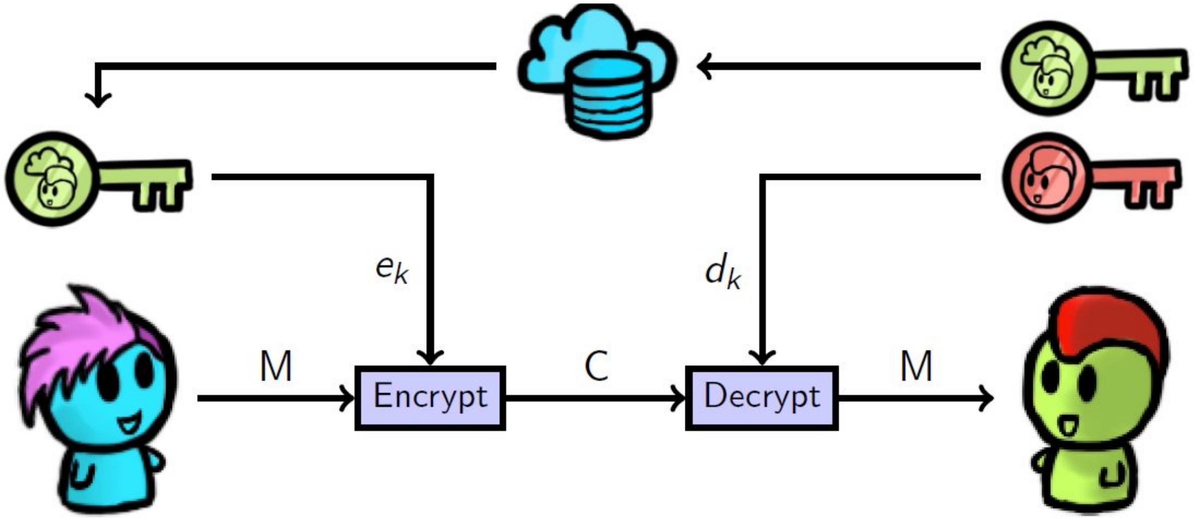
Cryptography Organization



Cryptography Organization



Public Key Cryptography, “1970s”



Examples:

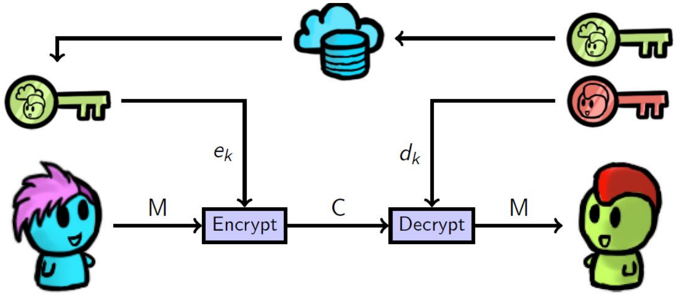
- RSA, ElGamal, ECC, NTRU

Steps for Public Key Cryptography?

1. Bob generates pair 

2. Bob gives everyone the public key 

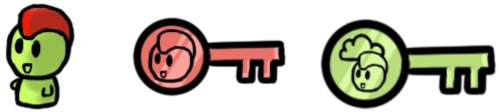
3. Alice encrypts m and sends it



4. Bob decrypts using private key

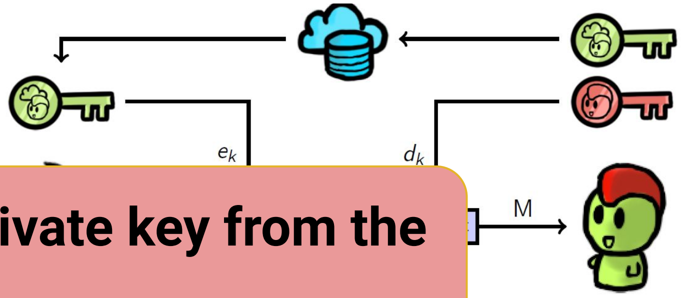
5. Eve and Alice can’t decrypt, only have encryption key

Steps for Public Key Cryptography?

1. Bob generates pair 

2. Bob gives everyone the public key 

3. Alice encrypts m and sends it



4. Bob d

It must be hard to derive the private key from the public key

5. Eve and Alice can't decrypt, only have encryption key

Requirements for PKE

- The encryption function? Must be easy to compute 
- The inverse, decryption? Must be hard for anyone without the key  vs. 

Thus, we require so called "one-way" functions for this.

Requirements for PKE

- The encryption function? Must be easy to compute 
- The inverse, decryption? Must be hard for anyone without the key  vs. 

Thus, we require so called “one-way” functions for this.

Because of encryption, also injective

Requirements for PKE

- The encryption function? Must be easy to compute 

- The inverse, decryption? Must be hard for anyone without the key  vs. 

Thus, we require so called “one-way” functions for this.

Because of encryption, also injective

Because of decryption, we need a “trapdoor”

Time for Textbook RSA

- Computational difficulty of the **factoring problem**
  - Given two large primes n = p*q, it is very hard to factor n.
- Modular arithmetic: integer numbers that “wrap around”
- Overview:

$$(m^e)^d \equiv m \pmod{n}$$

Easy for me to pick e, d, and n that satisfy that equation

Ugh. I know e and n (even m) and can’t find d!!!

Fun (?) Facts::

- RSA first popular public-key encryption method, published in 1977

Textbook RSA (Simplified Overview)

1. Choose two **"large primes"** *p* and *q* (secretly)
2. Compute $n = p*q$
3. "Choose" value e and find d such that $(m^e)^d \equiv m \pmod{n}$
4. **Public key**: (e, n)
5. **Private key**: d (other numbers tossed)
6. Encryption: $c \equiv m^e \pmod{n}$
7. Decryption: $c^d \pmod{n}$

Textbook RSA (Simplified Overview)

1. Choose two “**large primes**” *p* and *q* (secretly)
2. Compute n = p*q
3. “Choose” value e and find d such that $(m^e)^d \equiv m \pmod{n}$
4. **Public key**: (e, n)
5. **Private key**: d (other numbers tossed)
6. Encrypt
7. Decrypt

Decryption works, but factoring n breaks this!

Note:

- RSA? Rivest, Shamir, and Adleman

A Closer Look at RSA: Rec… Para…eters

1. Choose two **"large primes"**
2. Compute n = p*q
3. "Choose" valu…
4. **Public ke…**
5. **Privat…**)
6. Encryptio…
7. Decryp…n: c^d (r… d n)

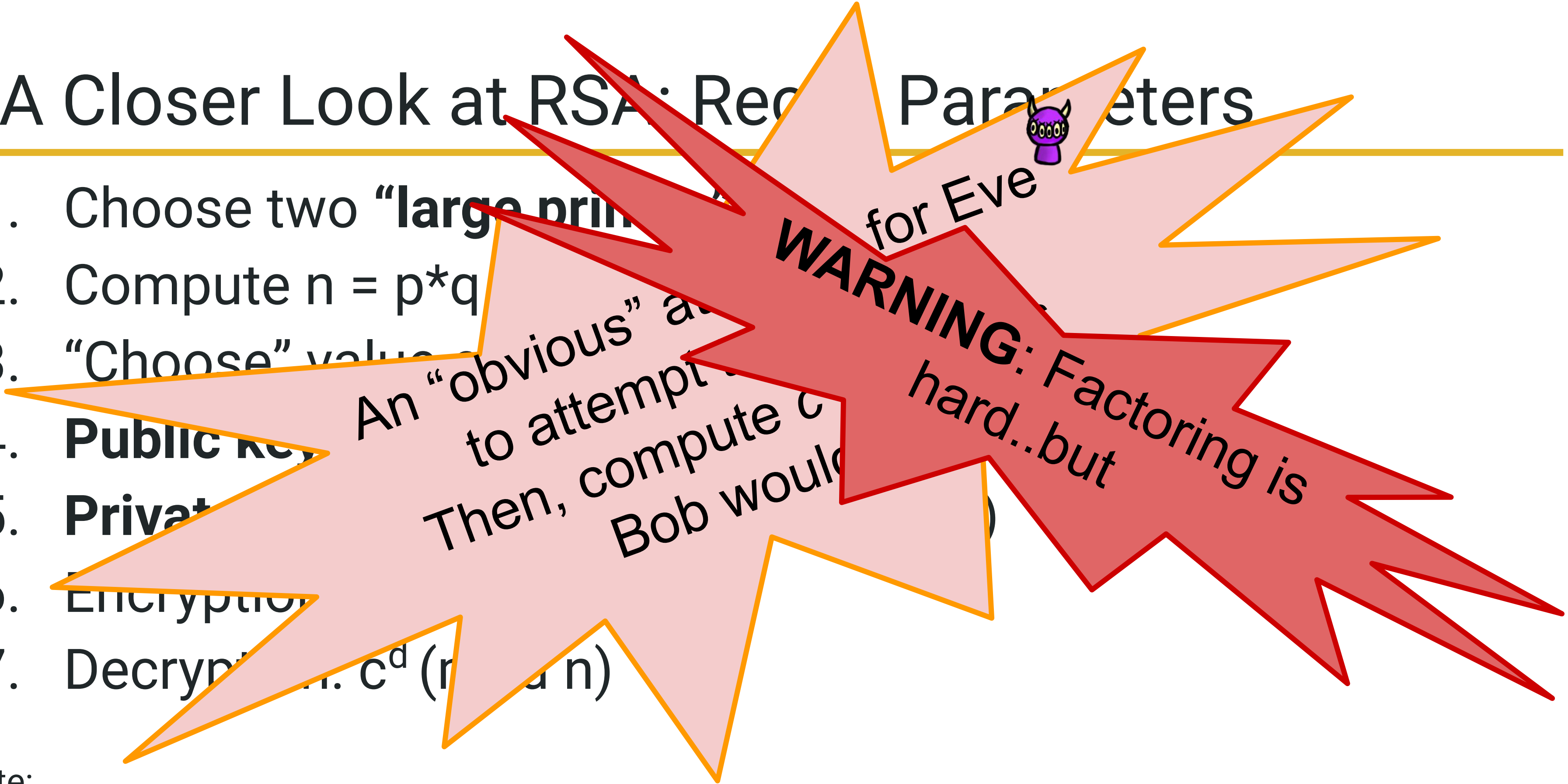
An "obvious" attack is for Eve
to attempt to factor *n*.
Then, compute c and e as
Bob would…

Note:

- RSA? Rivest, Shamir, and Adleman

A Closer Look at RSA: Rec Par eters

1. Choose two "large pri
2. Compute n = p*q
3. "Choose" valu
4. **Public ke**
5. **Privat**
6. Encryptio
7. Decryp n: c^d (r d n)



Note:

- RSA? Rivest, Shamir, and Adleman

Factoring and RSA

- You want to factor the public modulus? 
- Good news, abundant literature on factoring algorithms
- Bad news, “appropriate” primes will not be defeated

Bad primes: easily factored

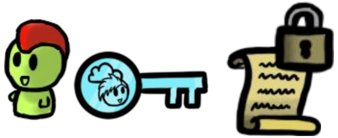
Malleability

$$\mathbf{A:}\ (m_1)^e * (m_2)^e = (m_1 * m_2)^e$$

It is possible to transform a ciphertext into another ciphertext that decrypts to a related plaintext

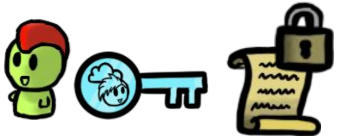
Undesirable (most of the time)

RSA and a Chosen Ciphertext Attack

- Alice is using RSA, public key (e, n) 
- Bob sends $c = E_e(m)$ 
- We are Eve! We snag c. 
- Alice…is confident about textbook RSA, will decrypt any ciphertext except c for us

Goal: Ask Alice to decrypt something (other than c) that helps us learn m

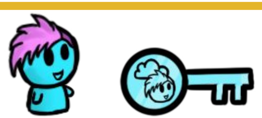
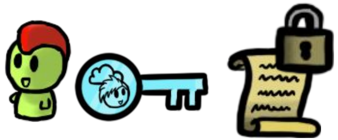
Executing CCA on Textbook RSA

- Alice is using RSA, public key (e, n) 
- Bob sends $c = E_e(m)$ 
- We-Eve ask Alice to decrypt $c_2 = 2^e * c_1$

Q: Decrypts to?

Q: Decrypts to?

Executing CCA on Textbook RSA

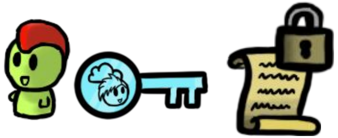
- Alice is using RSA, public key (e, n) 
- Bob sends $c = E_e(m)$ 
- We-Eve ask Alice to decrypt $c_2 = 2^e * c_1$ 

Q: Decrypts to?

Q: Decrypts to?

A: decryption gives $(2^e * c_1)^d \equiv 2m$

Executing CCA on Textbook RSA

- Alice is using RSA, public key (e, n) 
- Bob sends $c = E_e(m)$ 
- We-Eve ask Alice to decrypt $c_2 = 2^e * c_1$ 

I am so clever mwahaha

Q: Decrypts to?

A: decryption gives $(2^e * c_1)^d \equiv 2m$

Textbook RSA: vulnerable to CCA
Note: Can be addressed with padding techniques

Show Naive RSA Encryption is not IND-CPA Secure

1. Eve produces two plaintexts, m_0 and m_1 

Show Naive RSA Encryption is not IND-CPA Secure

1. Eve produces two plaintexts, m_0 and m_1 

2. "Challenger" encrypts an m as $c^* \leftarrow m_b^e \pmod{N}$, secret b 

Show Naive RSA Encryption is not IND-CPA Secure

1. Eve produces two plaintexts, m_0 and m_1 

2. "Challenger" encrypts an m as $c^* \leftarrow m_b^e \pmod{N}$, secret b 

3. Eve's goal? Determine $b \in \{0,1\}$

Show Naive RSA Encryption is not IND-CPA Secure

1. Eve produces two plaintexts, m_0 and m_1 

2. "Challenger" encrypts an m as c^* <- m_b^e (mod N), secret b 

3. Eve's goal? Determine $b \in \{0,1\}$

4. Sooo, Eve computes c <- m_1^e (mod N)

   If $c^* = c$ then Eve knows $m_b = m_1$
   If c^* <> c then Eve knows $m_b = m_0$

Show Naive RSA Encryption is not IND-CPA Secure

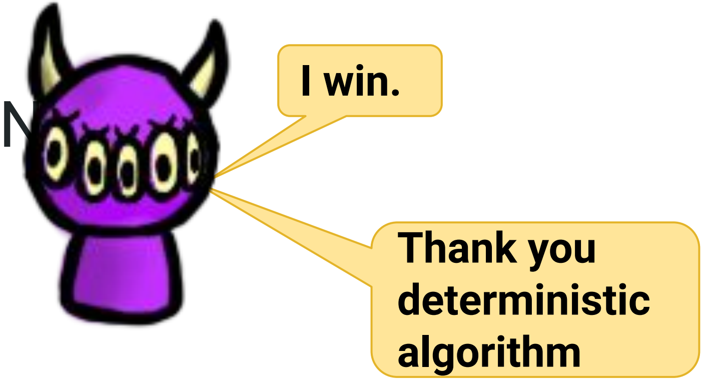
1. Eve produces two plaintexts, m_0 and m_1 

2. “Challenger” encrypts an m as c^* <- m_b^e (mod N), secret b 

3. Eve’s goal? Determine $b \in \{0,1\}$

4. Sooo, Eve computes c <- m_1^e (mod N

   If c^* = c then Eve knows $m_b = m_1$
   If c^* <> c then Eve knows $m_b = m_0$



Adversaries and their Goals

You've assumed my goal is the secret/private key...

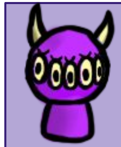
Adversaries and their Goals

You've assumed my goal is the secret/private key…

…but less ambitious goals can be very effective…

Adversaries and their Goals

Goal 1: Total Break

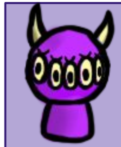


- Win the secret key k or
- Win Bob's private key k_b
- Can decrypt any c_i *for:*

$$c_i = E_k(m) \text{ or } c_i = E_{kb}(m)$$

- All messages using compromised k revealed
- Unless **detected** game over

Goal 2: Partial Break

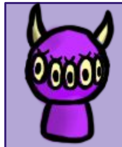


- Decrypt **a ciphertext** c (without the key)
- Learn **some** specific information about a message m from c

**Need to occur with non-negligible probability.

- **Some (or a)** message revealed

Goal 3: Distinguishable Ciphertexts



- *P{learn b* ∈ {0,1}} exceeds ½
- Distinguish between $E(m_1)$ and $E(m_2)$ or between E(m) and E(random string)

- The ciphertexts are leaking small/some information…

Until next time...