Engineering Robust Server Software

Cryptography

Significant portions based on slides from
Micah Sherr @ Georgetown

Brian Rogers Duke ECE

Cryptography

f(Leftover Food in HH 218)

= Al481manj417a@#1naL

f^{-1}(Al481manj417a@#1naL)

=Leftover Food in HH 218

Alice

Al481manj417a@#1naL

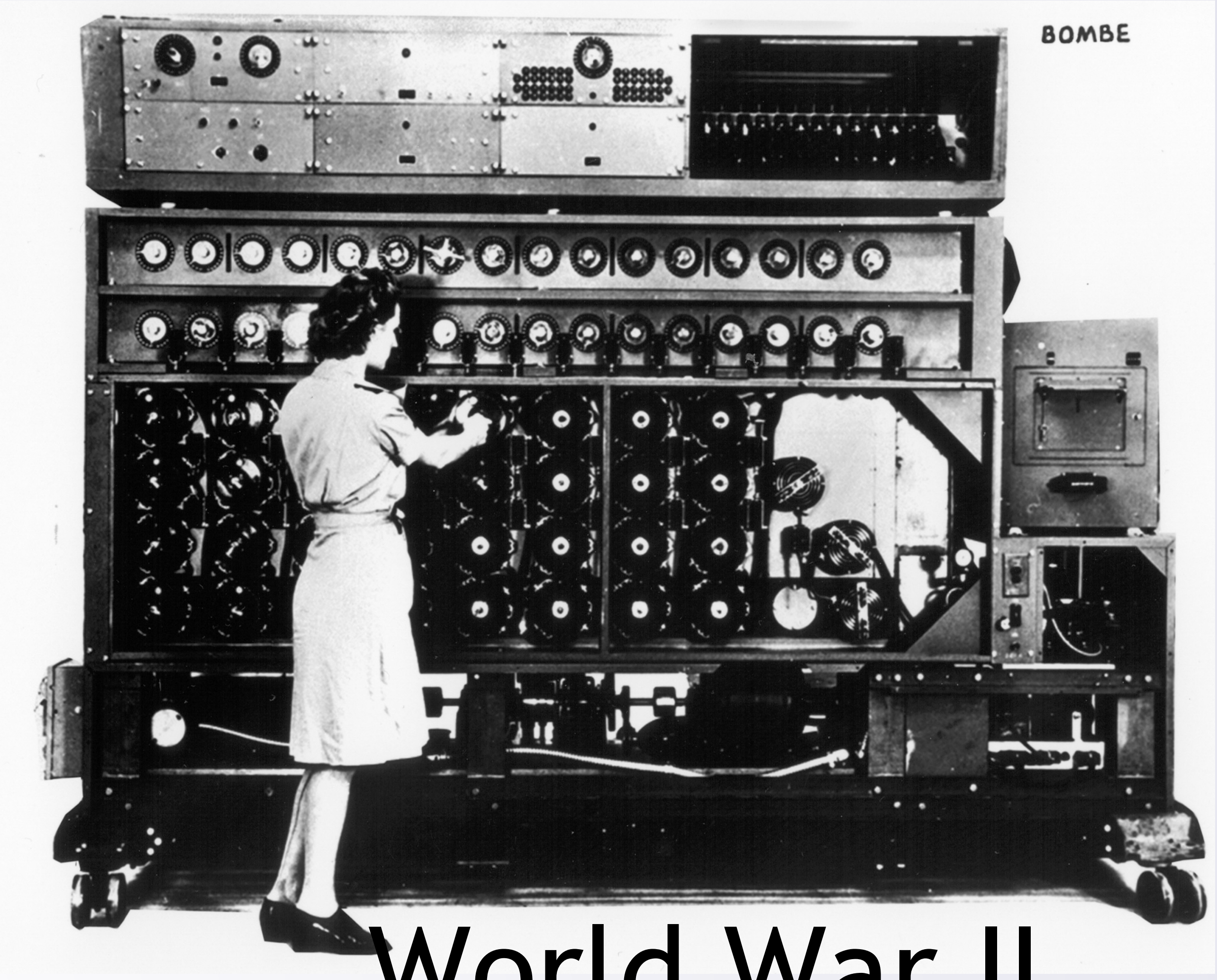
Bob

??

Eve

This is an example of..
A: Confidentiality
B: Integrity
C: Authentication
D: Availability

Ancient History to Modern Times



Mesopotamia
~1500 BCE

Caesar Cipher

World War II

Egypt
~1900 BCE

Roman Empire
~80 BCE

Vigenère Cipher
1553

AES/RSA
Present

- Modern cryptography: secure; advanced math
- Classical cryptography: insecure; simple math

Cryptography Terms

Plaintext

```
FIRST LEGION
ATTACK EAST
FLANK
```

Encrypt

E_k(Plaintext)

Ciphertext

```
ILUVW OHJLRQ
DWWDFN HDVW
IODQN
```

Decrypt

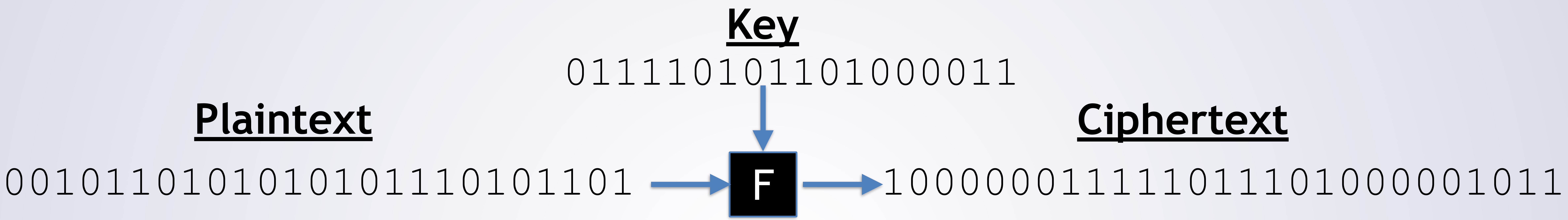
D_k(Ciphertext)

- **Cryptosystem**: method of disguising (encrypting) plaintext messages so that only select parties can decipher (decrypt) the ciphertext
- **Cryptography**: the art/science of developing and using cryptosystems
- **Cryptanalysis**: the art/science of breaking cryptosystems
- **Cryptology**: the combined study of cryptography and cryptanalysis

Kerckhoffs' Principles

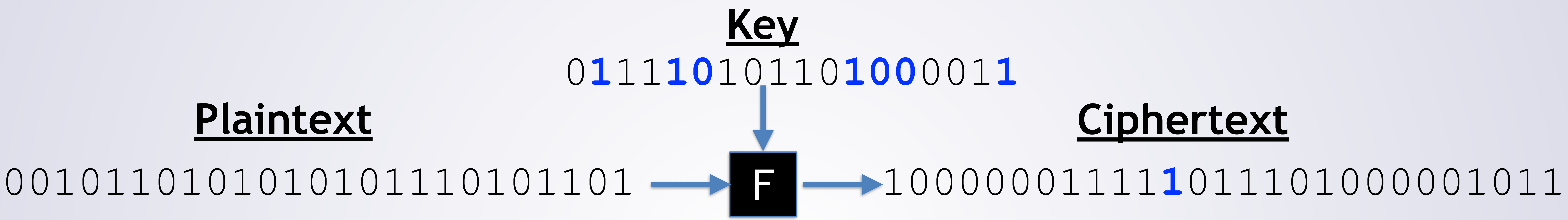
- Kerckhoffs' principles [1883]:
  - Assume Eve knows cipher algorithm
  - Security should rely on choice of key
  - If Eve discovers the key, a new key can be chosen
- Opposite of "security by obscurity"
  - Idea of keeping algorithm secret
- Why not security by obscurity?
  - Compromised? Destroyed. (vs one key lost-> make new one)
  - Algorithms relatively easy to reverse engineer

Shannon's Principles



- Two important principles for modern/practical systems:
  - Confusion: each bit of cipher text depends on many key bits
  - Diffusion: flipping one bit of plaintext should alter many (~½) of ciphertext

Shannon's Principles



- Two important principles for modern/practical systems:
  - **Confusion**: each bit of cipher text depends on many key bits
  - Diffusion: flipping one bit of plaintext should alter many (~½) of ciphertext

Shannon's Principles

Key

0111101011010000011

Plaintext → F → **Ciphertext**

00101101010101011101011101 → 100000011111011110100000101 1

00101101010111011101011101 → 010101101101101101100101010

- Two important principles for modern/practical systems:
  - Confusion: each bit of cipher text depends on many key bits
  - **Diffusion**: flipping one bit of plaintext should alter many (~½) of ciphertext

Classical Cryptography

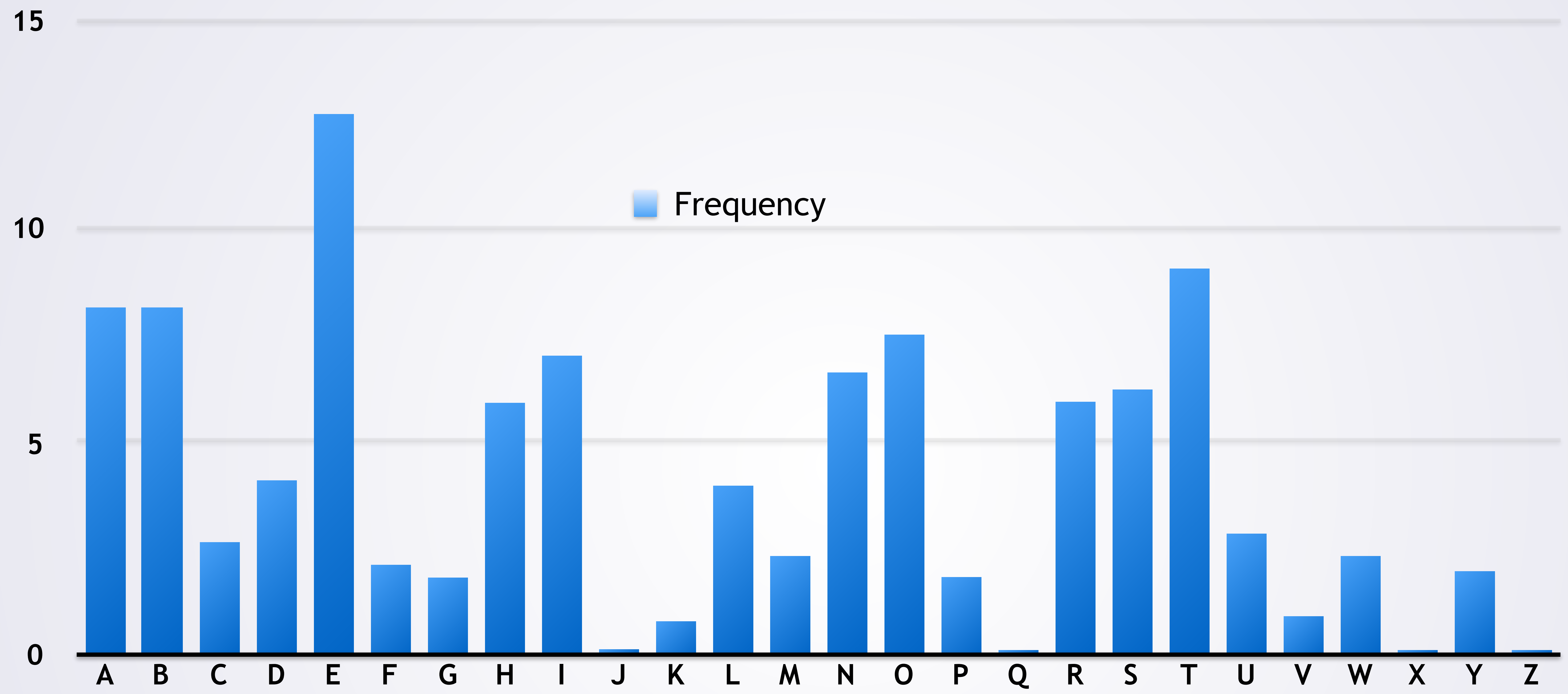
FIRST LEGION ATTACK EAST FLANK

+3 ↓

ILUVW OHJLRQ DWWDFN HDVW IODQN

- Simple/ancient classical crypto system:
  - Caesar Cipher: named after Julius Caesar
- **Key**: number of letters to shift by (in this case 3)

Breaking Caesar



- You may have previously written a program to crack this
  - 'e' is most common in English
  - Find most common in ciphertext -> probably 'e'

Spaces and Punctuation

FIRST LEGION ATTACK EAST FLANK

ILUVW OHJLRQ DWWDFN HDVW IODQN

- Quick side note:
  - I'm writing spaces in the plain text/cipher text (readability of examples)
  - Would not really do (makes much easier)
- Either encrypt spaces/punctuation too (computers) or
- Remove from plaintext before encrypting

Vigenère Cipher

```
FIRST LEGION ATTACK EAST FLANK
drago ndrago ndrago ndra godra
```

+3 +17 +0

```
IZRYH OVGOCA RTZOPN EGGG WLGBX
```

- Key is now a vector of numbers , e.g., (3,17,0,6,14,13)
  - Usually represented by a word "dragon"

Vigenère Security?

- Vigenère seemed unbreakable for a few centuries
  - Long enough key: smooth out frequencies
- Easy to break if you can determine key length
  - Key length 10?
    - Take letters 0, 10, 20,… frequency count
    - 1, 11, 21, 31, … frequency count. etc.
- Try many different key lengths?
  - Time consuming with pencil and paper
  - Easy with computer…
  - Vigenère broken even before computers

Vigenère

- Vigenère is what many novices make up on their own
  - Seems hard to break!
  - …but is actually easy.
- Important lesson:
  - Do not try to make up your own crypto
  - It is very hard to do correctly
- But what if…
  - Your key were as long as your message
  - And you only used it for one message?

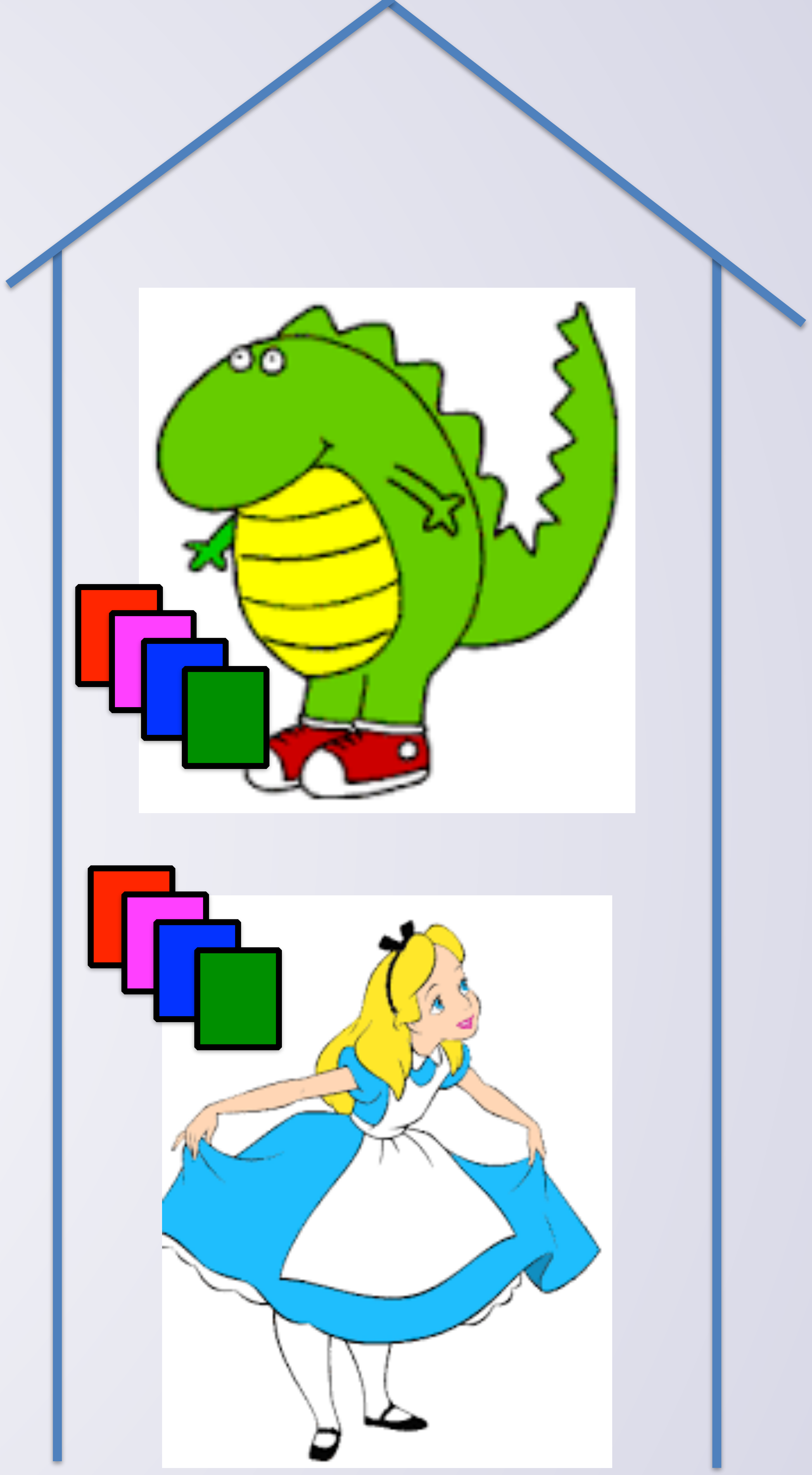
One Time Pad

- One Time Pad
  - $E_k(M) = M \oplus K$
  - Length of K is equal to Length of M (same number of bytes)
  - NEVER re-use K
    - Re-using even once destroys guarantees
- Gives perfect secrecy
  - Without knowledge of key, guessing M is just random guessing
- Difficult in practice
  - Must exchange keys securely, and cannot re-use

One Time Pad

Alice and Bob are in HQ

They generate some OTPs

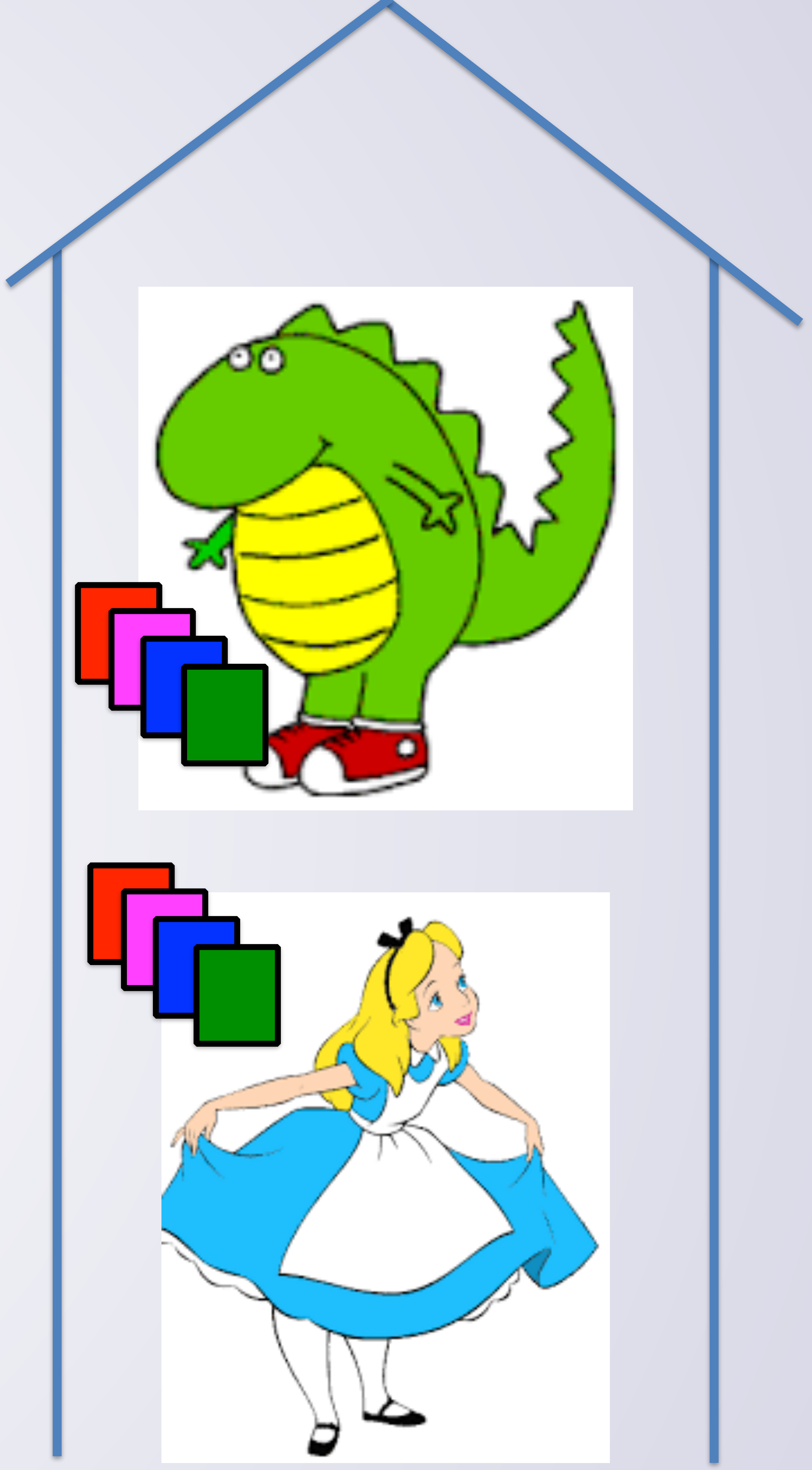


One Time Pad

Alice and Bob are in HQ

They generate some OTPs

Now, Alice goes into the field

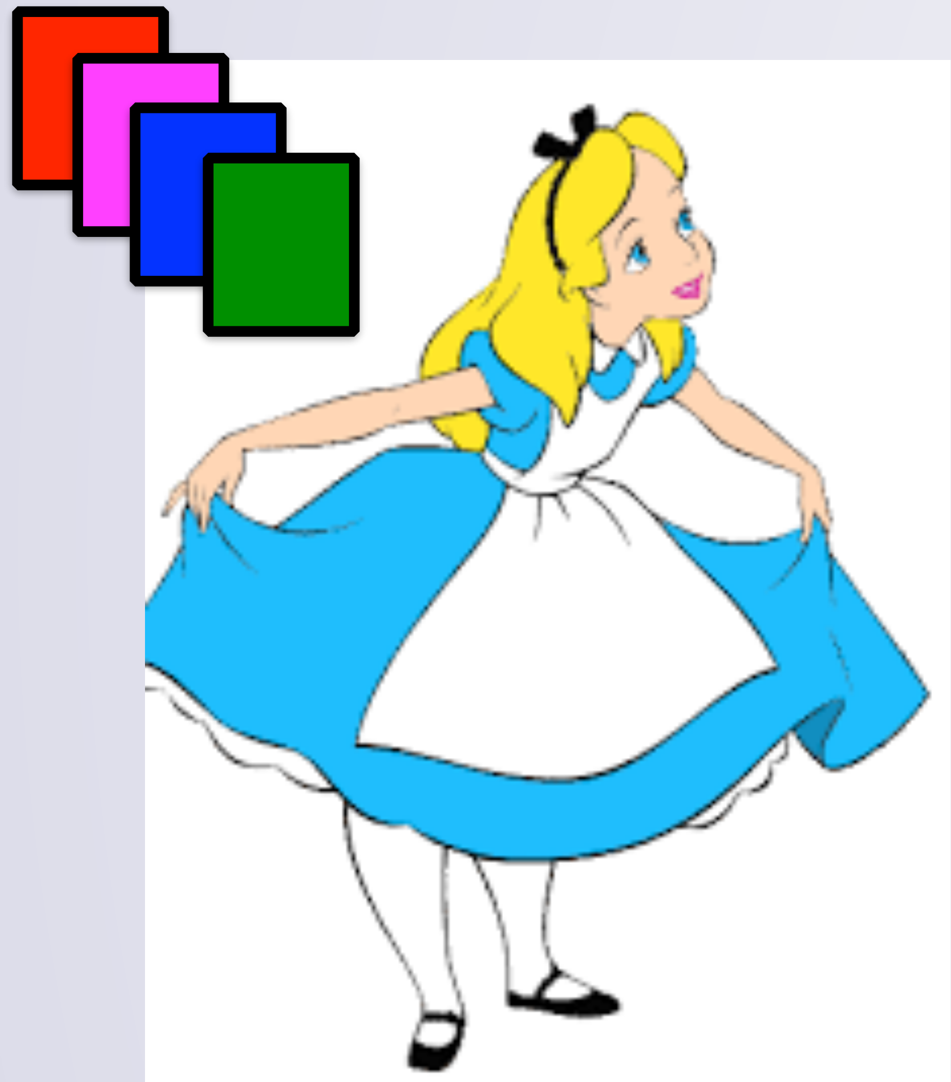


Duke UNIVERSITY

One Time Pad

$C_1 = M_1 \oplus K_1$

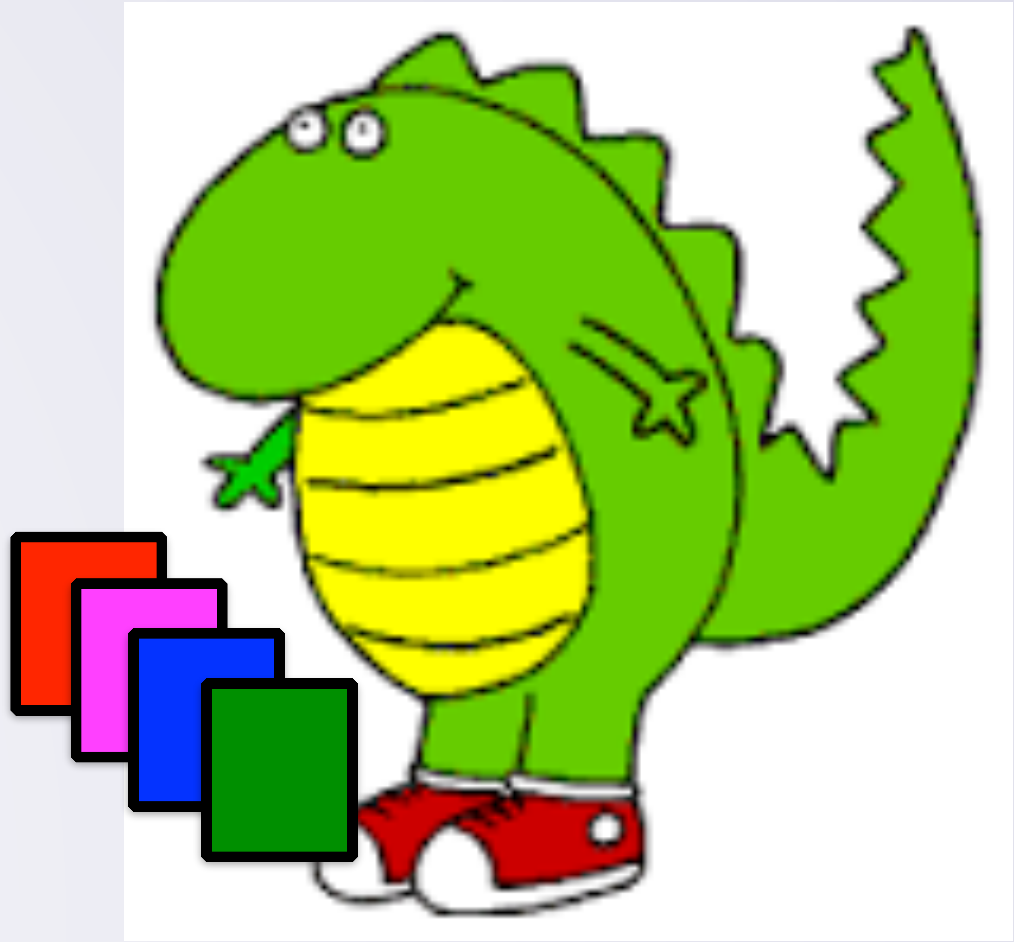
$M_2 = C_2 \oplus K_2$



Alice and Bob are in HQ

They generate some OTPs

Now, Alice goes into the field

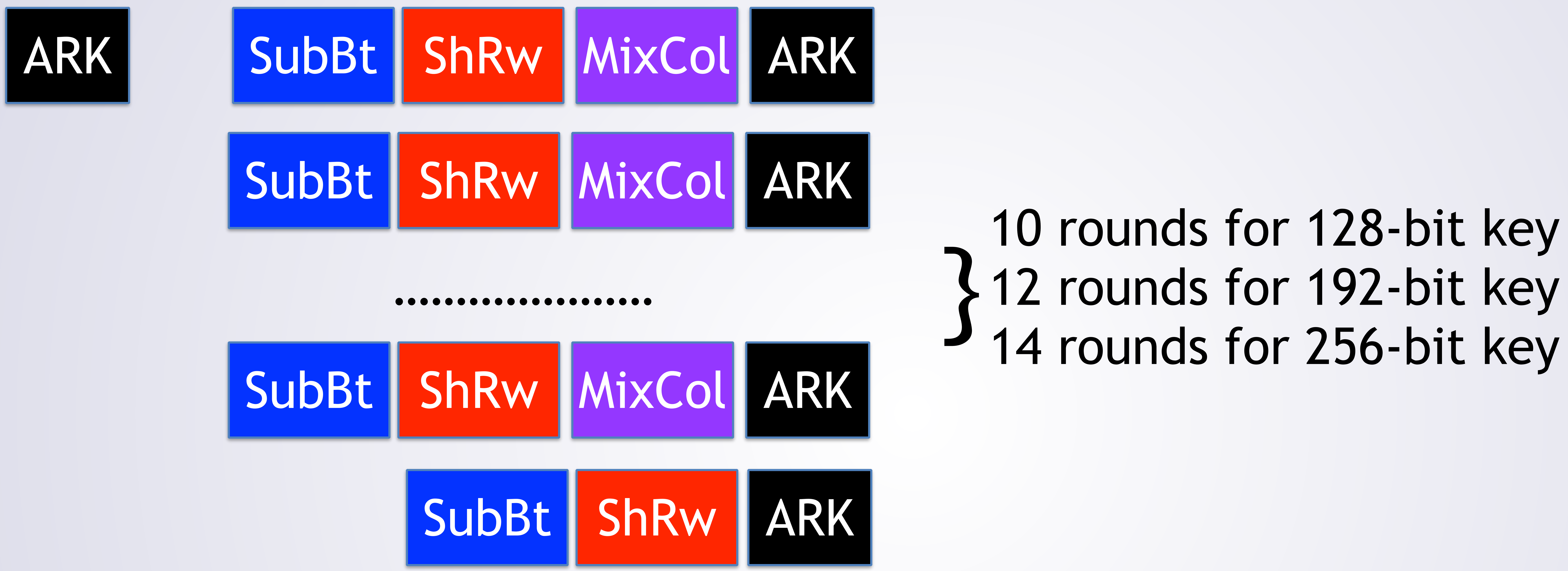


$M_1 = C_1 \oplus K_1$

$C_2 = M_2 \oplus K_2$

??

AES



- Advanced Encryption Standard (Rijndael)
  - Symmetric key (Alice and Bob have same key)
  - Replaced DES as accepted symmetric key standard block cipher
  - **"Nobody ever got fired for using AES"**

AES: Add Round Key

Input

00	01	02	03
10	11	12	13
20	21	22	23
30	31	32	33

$\oplus$

Round key

1F	3C	09	AB
2C	D9	11	AA
FC	00	99	21
38	8E	07	4C

=

Output

1F	3D	0B	A8
3C	C9	03	B9
DC	21	BB	02
08	BF	35	7F

- Add Round Key ARK
    - XOR input data with **round key**
- What is a round key?
    - At the start, key is expanded into 11 (13, or 15) round keys
    - Each round key is used once

AES: Substitute Bytes

Input

00	01	02	03
10	11	12	13
20	21	22	23
30	31	32	33

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
00	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
10	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
20	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
30	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
40	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
50	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
60	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
70	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
80	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
90	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a0	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b0	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c0	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
[illegible]	[illegible]	[illegible]	[illegible]	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
[illegible]	[illegible]	[illegible]	[illegible]	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
[illegible]	[illegible]	[illegible]	[illegible]	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

SubBt

Output

63	7C	77	7B
CA	82	C9	7D
B7	FD	93	26
04	C7	23	7F

- Substitute Bytes
  - Look up input in substitution table ("sbox").
  - Substitution is 1-to-1 (each value appears once in the table)
  - AES's Sbox designed with important mathematical properties

AES: Shift Rows

Input

00	01	02	03
10	11	12	13
20	21	22	23
30	31	32	33

Output

00	01	02	03
11	12	13	10
22	23	20	21
33	30	31	32

- Shift Rows ShRw
  - Shift the (ith) row left by i positions
  - Row 0: no change
  - Row 1: shift bytes left one poition

AES: Mix Columns

Input

00	01	02	03
10	11	12	13
20	21	22	23
30	31	32	33

$$[01\ 11\ 21\ 31] * \begin{matrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{matrix}$$

Output

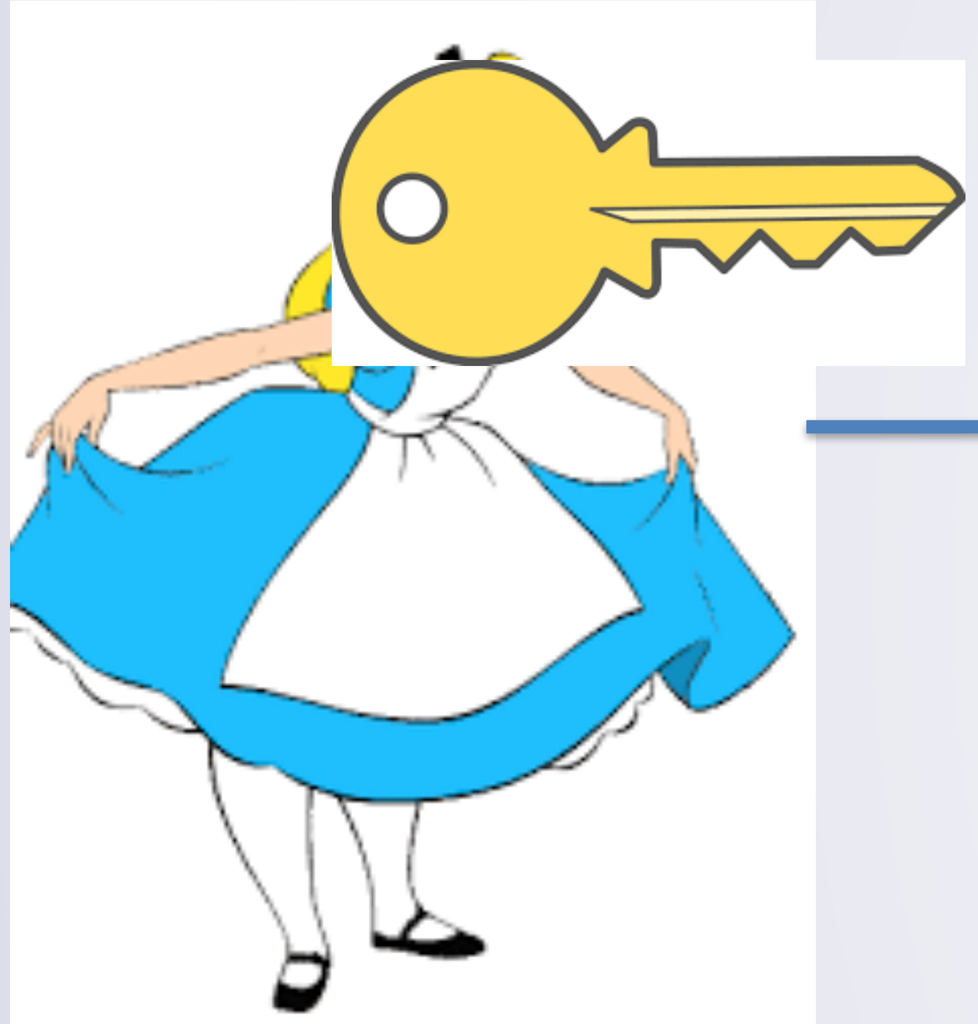
	XX		
	XX		
	XX		
	XX		

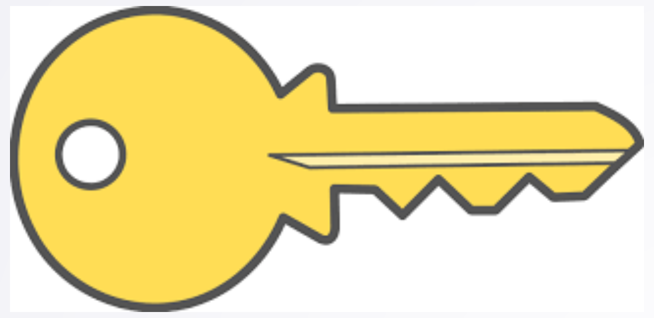
- Mix Columns   MixCol
  - Take each input column
  - Multiply it by a matrix as polynomial in GF(2^8)
  - Result is column in output

AES: Confusion and Diffusion?

- Does AES have good confusion and diffusion?
  - **A**: Good confusion and good diffusion
  - **B**: Good confusion, poor diffusion
  - **C**: Poor diffusion, good confusion
  - **D**: Poor diffusion and poor confusion

Difficulty: Key Distribution





Bwahahah!

Diffie-Hellman Key Exchange

$S = B^x \bmod p$

$A = g^x \bmod p$

Secret: x

$S = A^y \bmod p$

$B = g^y \bmod p$

Secret: y

Here are two prime numbers: g and p →

Here is the value of A →

← Here is the value of B

I also know g and p

I also know A

I also know B

Diffie-Hellman Key Exchange

$S = B^x \bmod p$

$A = g^x \bmod p$

Secret: x

Alice:

$S=(g^y \bmod p)^x \bmod p$

Bob:

$S=(g^x \bmod p)^y \bmod p$

These are equal

$S = A^y \bmod p$

$B = g^y \bmod p$

Secret: y

Eve has to solve
the discrete logarithm **(hard)**
problem to recover x or y
(and thus compute S)

I also know **g** and **p**

I also know **A**

I also know **B**

Diffie-Hellman Key Exchange

$S = B^x \bmod p$

$A = g^x \bmod p$

Secret: x

This scheme works securely as long as…

A: …Alice and Bob pre-share at least lg(p) bits

B: …p is at least 64 bits

C: …Eve can only listen, not alter messages

D: …Eve does not have many GPUs

$S = A^y \bmod p$

$B = g^y \bmod p$

Secret: y

I also know **g** and **p**

I also know **A**

I also know **B**

Diffie-Hellman Key Exchange

$S = B^x \bmod p$

$A = g^x \bmod p$

Secret: x

$S = A^y \bmod p$

$B = g^y \bmod p$

Secret: y

All of this assume Eve can only **listen**.
What if Eve can **change** the messages?

I also know **g** and **p**

I also know **A**

I also know **B**

Man In the Middle (MITM) Attack

$S = C^x \bmod p$

$A = g^x \bmod p$

Secret: x

$S = C^y \bmod p$

$B = g^y \bmod p$

Secret: y

Here are two numbers: g and p

Here is the value of A

(Replace A with C)

(Replace B with C)

Here is the value of B

I also know g and p

I also make: z

$C = g^z \bmod p$

$S_{Alice} = A^z \bmod p$

$S_{Bob} = B^z \bmod p$

Man In the Middle (MITM) Attack

$S = C^x \bmod p$

$A = g^x \bmod p$

Secret: x

At this point, Eve has exchanged (different) keys with Alice and Bob.

Eve can now decrypt, view (and alter) a message, then encrypt it and send it along.

$S = C^y \bmod p$

$B = g^y \bmod p$

Secret: y

I also know g and p

I also make: z

$C = g^z \bmod p$

$S_{Alice} = A^z \bmod p$

$S_{Bob} = B^z \bmod p$

Man In The Middle Attack

- Alice needs to know that she is receiving Bob's message unchanged
  - Which security principles are these?
    - A: Confidentiality and Integrity
    - B: Integrity and Authentication
    - C: Authentication and Availability
    - D: Integrity and Confidentiality

Man In The Middle Attack

- Alice needs to know that she is receiving Bob's message unchanged
  - Which security principles are these?
- **Integrity**: don't let Eve tamper with things
- **Authentication**: message actually came from Bob (not someone else)
- Cryptographic solution: **signatures**
  - Bob will generate a cryptographic validation of the message
  - (and that it was from him)
- For this, we need **public key** cryptography: e.g., RSA
  - Also called **asymmetric key** cryptography

Public Key Cryptography

Bob picks two random primes: p and q

Bob computes n = pq

Number of bits in n is key length

Bob computes $\lambda(n) = \mathrm{lcm}(p-1, q-1)$

Bob picks e st. $1 < e < \lambda(n)$

- Where e and $\lambda(n)$ are coprime

Bob solves for $d = e^{-1} \bmod \lambda(n)$

- That is $ed = 1 \bmod \lambda(n)$

Bob publishes his **public key** (e,n)

Bob keeps private key d, secret (he also keeps n)

Encryption

Bob computes $M = C^d \bmod n$

Bob's public key: (e,n)

Private key: (d,n)

Alice can send a message (M) to Bob:
She computes **$C = M^e \bmod n$**
and sends C to Bob

Eve does not have **d**.
She cannot recover the message from (e,n)

Bob's public key: (e,n)

Signing

Alice computes $M' = S^e \bmod n$
checks that $M = M'$

Bob's public key: (e,n)

Private key: (d,n)

Bob wants Alice to know message M is from him. He computes
$S = M^d \bmod n$
and sends M and S to Alice

Eve cannot fake this signature because she does not have **d**

Bob's public key: (e,n)

Duke UNIVERSITY

...But Did We Fix Anything?

Great if Alice can get Bob's public key in a trusted way
(then again, she could get an AES key that way)... but if not...?

Bob publishes his **public key** (e,n)

(e,n)

(e_{fake}, n_{fake})

What if Eve intercepts and convinces Alice of a fake key?

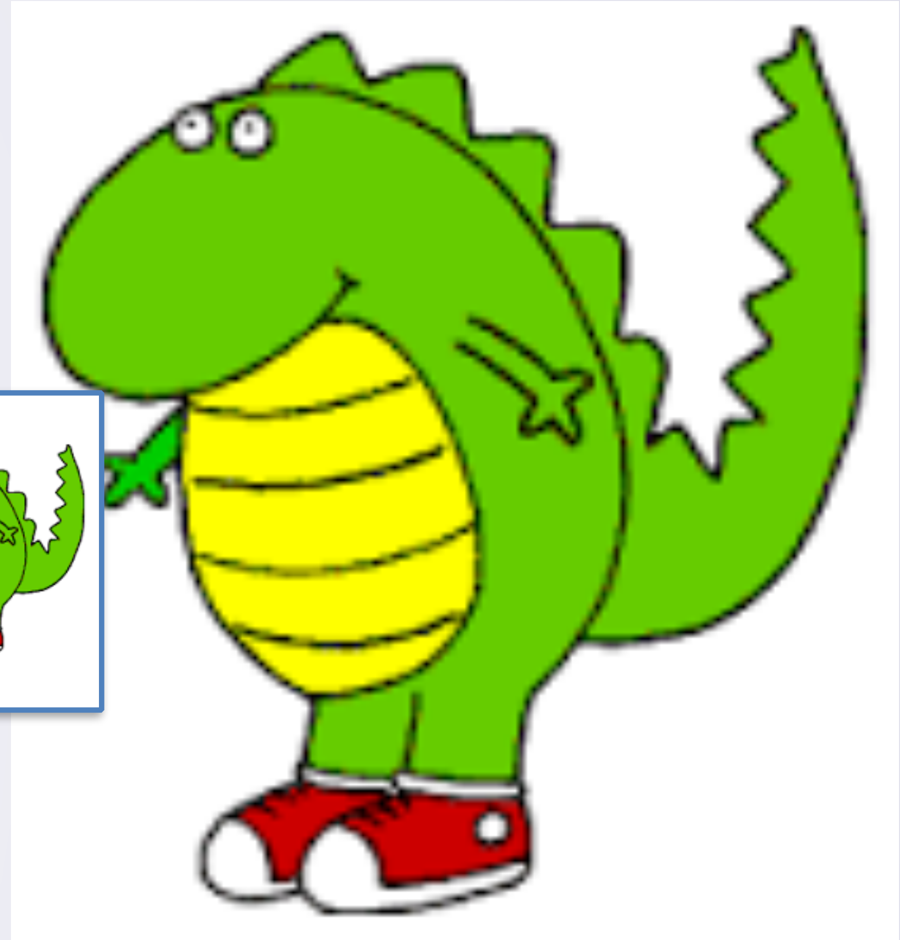
...But Did We Fix Anything?

Suppose Alice already trusts Ted.

Ted's public key

(e,n)

Bob T. Dino
Scales: Grn
Tail: spiked



Bob can take his public key (and proof that he is Bob) to Ted

Now Bob can send the signed key to Alice

Ted can sign Bob's key: he can make the message "Bob's key is (e,n)" and cryptographically sign it with his key

Bob's key is (e,n)
— Ted

Certificates

- Certificates: electronic documents attesting to ownership of a key
  - Cryptographically signed by Certificate Authority (CA)
    - To be meaningful, CA needs to be trusted
    - Trust may be done in several steps: A signs B, B signs C.
  - Generally contains expiration date
- https uses certificates
  - Your computer trusts certain CAs

Side Channels

- AES + RSA: hard to break algorithmically
  - VERY Difficult to recover key, or decipher message without key
- Can be attacked by **side channels**
  - Information leaked from physical characteristics of execution
  - E.g., power, temperature, memory access pattern, instruction timing…

Side Channel Example

- AES: some steps sped up with 4KB lookup table
  - Indexed by input to that stage
  - Tell which cache block -> gain much information -> recover key
- Attacker runs code on same core
  - Measures time to perform loads
  - Determines hits/misses in cache
  - Figures out "victim"'s memory access pattern
- Similar attacks on RSA based on multiplication patterns
  - Timing, power, …

Cryptography Wrap Up

- Quick introduction to basics of cryptography
  - Classical systems: weak
  - AES: symmetric key
  - RSA: public key (asymmetric key)
- A few attacks:
  - MITM
  - Side channels
- Idea of signing + certificates