

Cryptography: Concepts & Confidentiality

A photograph of Taylor Swift sitting in a dark, ornate chair. She is wearing a strapless, heavily embellished silver dress that appears to be covered in crystals or sequins. She is also wearing multiple pieces of jewelry, including a large diamond necklace, a matching bracelet, and several rings. Her hair is styled in loose waves, and she has bright red lipstick on. She is looking off to the side with a serious expression. The background is dark and out of focus, showing some architectural details. Overlaid on the lower half of the image is a block of text in a bold, red, sans-serif font.

**Cryptography is nightmare magic
math that cares what kind of pen
you use -@swiftonsecurity**

Cryptography: Philosophy...

- This part of the class is really ***don't try this at home***
 - It is ***incredibly easy*** to screw this stuff up
- It isn't just a matter of making encryption algorithms...
 - Unless your name is David Wagner or Raldua Popa, ***your crypto is broken!***
- It isn't just a matter of coding good algorithms...
 - Although just writing 100% correct code normally is hard enough!
- There is all sorts of deep voodoo that, ***when*** you screw up your security breaks
 - EG, bad random number generators, side channel attacks, reusing one-use-only items, replay attacks, downgrade attacks, you name it...



Three main goals

- ***Confidentiality***: preventing adversaries from ***reading*** our private data
 - Data = message or document
- ***Integrity***: preventing attackers from ***altering*** our data
 - Data itself might or might not be private
- ***Authentication***: proving who ***created*** a given message or document
 - Generally implies/requires integrity

Special guests

- Alice (sender of messages)
- Bob (receiver of messages)
- The attackers
 - Eve: “eavesdropper”
 - Mallory: “manipulator”



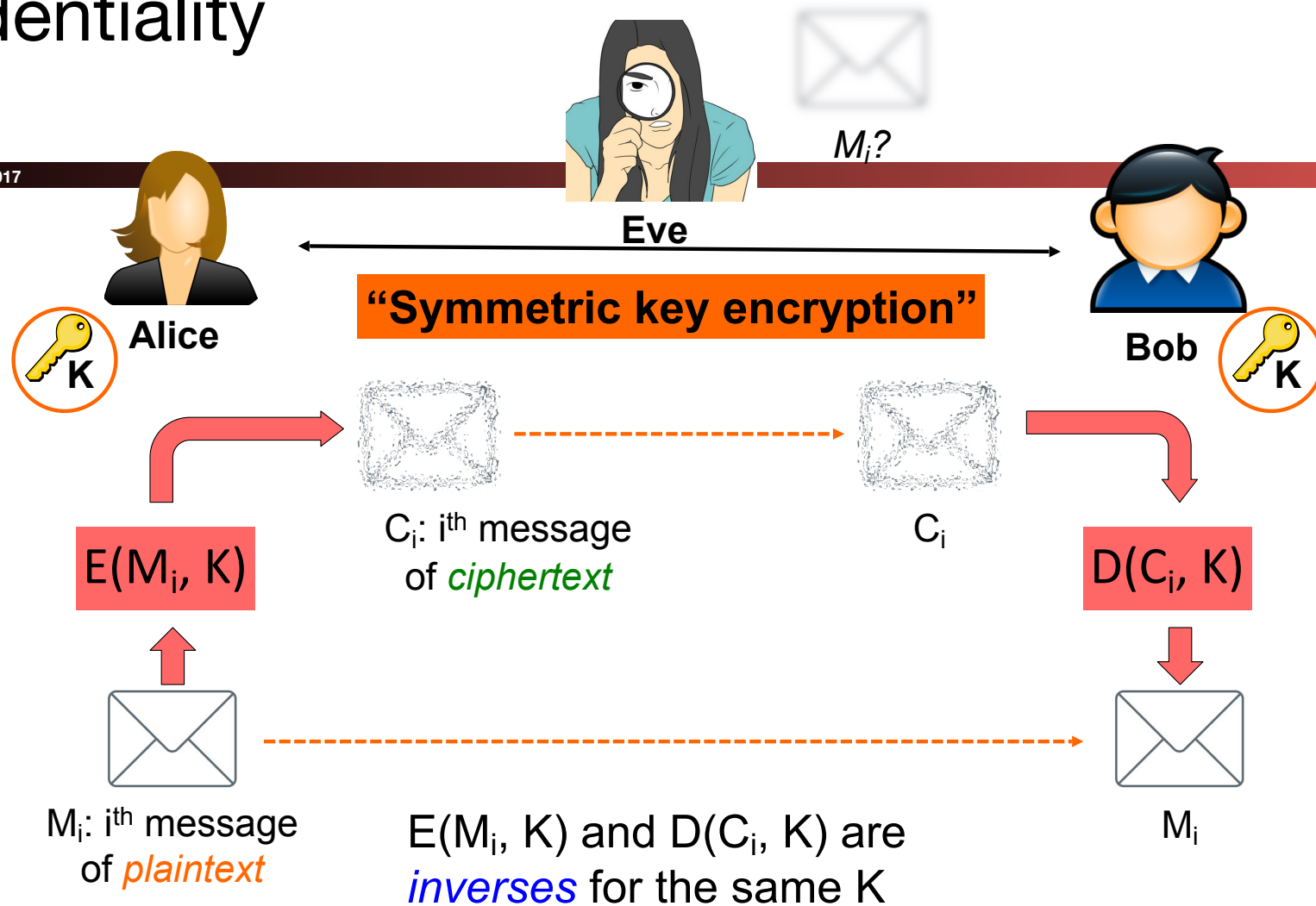
Eve



Confidentiality

Computer Science 161 Fall 2017

Weaver



The Ideal Contest

- Attacker's goal: any knowledge of M_i beyond an upper bound on its length
 - Slightly better than 50% probability at guessing a single bit: attacker wins!
 - Any notion of how M_i relates to M_j : attacker wins!
- Defender's goal: ensure attacker has no reason to think any $M' \in \{0,1\}^n$ is more likely than any other
 - (for M_i of length n)

Eve's Capabilities/Foreknowledge

- No knowledge of **K**
 - We assume **K** is selected by a *truly random process*
 - For **b**-bit key, any **K** $\in \{0,1\}^b$ is equally likely
- Recognition of success: Eve can generally tell if she has correctly and fully recovered **M_i**
 - But: Eve cannot recognize anything about *partial solutions*, such as whether she has correctly identified a particular bit in **M_i**
 - There are some attacks where Eve can guess and verify
 - Does not apply to scenarios where Eve exhaustively examines every possible **M_i'** $\in \{0,1\}^n$

Eve's Available Information

1. Ciphertext-only attack:

- Eve gets to see every instance of C_i
- Variant: Eve may also have partial information about M_i
 - “It’s probably English text”
 - Bob is Alice’s stockbroker, so it’s either “Buy!” or “Sell”

2. Known plaintext:

- Eve knows part of M_i and/or entire other M_j s
- How could this happen?
 - Encrypted HTTP request: starts with “GET”
 - Eve sees earlier message she knows Alice will send to Bob
 - Alice transmits in the clear and then resends encrypted
 - Alex the Nazi always transmits the weather report at the same time of day, with the word “Wetter” in a known position



Eve's Available Information, con't

3. Chosen plaintext

- Eve gets Alice to send \mathbf{M}_j 's of Eve's choosing
- How can this happen?
 - E.g. Eve sends Alice an email spoofed from Alice's boss saying "Please securely forward this to Bob"
 - E.g. Eve has some JavaScript running in Alice's web browser that is contacting Bob's TLS web server

4. Chosen ciphertext:

- Eve tricks Bob into decrypting some \mathbf{C}_j of her choice and he reveals something about the result
- How could this happen?
 - E.g. repeatedly send ciphertext to a web server that will send back different-sized messages depending on whether ciphertext decrypts into something well-formatted
 - Or: measure how long it takes Bob to decrypt & validate



Eve's Available Information, con't

5. Combinations of the above

- Ideally, we'd like to defend against this last, the most powerful attacker
- And: we can!, so we'll mainly focus on this attacker when discussing different considerations



Independence Under Chosen Plaintext Attack game: IND-CPA

- Eve is interacting with an encryption "Oracle"
 - Oracle has an unknown random key k
- She can provide two separate chosen plaintexts of the same length
 - Oracle will randomly select one to encrypt with the unknown key
 - The game can repeat...
- Goal of Eve is to have a better than random chance of guessing which plaintext the oracle selected
 - Variations involve the Oracle always selecting either the first or the second

Designing Ciphers

- Clearly, the whole trick is in the design of **$E(M,K)$** and **$D(C,K)$**
- One very simple approach:
 $E(M,K) = \text{ROTK}(M)$; $D(C,K) = \text{ROT-K}(C)$
i.e., take each letter in **M** and “rotate” it **K** positions (with wrap-around) through the alphabet
- E.g., **$M_i = \text{“DOG”}$** , **$K = 3$**
 $C_i = E(M_i,K) = \text{ROT3(“DOG”)} = \text{“GRJ”}$
 $D(C_i,K) = \text{ROT-3(“GRJ”)} = \text{“DOG”}$
- “Caesar cipher”
 - "This message has been encrypted twice by ROT-13 for your protection"



Attacks on Caesar Ciphers?

- Brute force: try every possible value of K
 - Work involved?
 - At most 26 “steps”

Attacks on Caesar Ciphers?

- Brute force: try every possible value of K
 - Work involved?
 - At most 26 “steps”
- Deduction:
 - Analyze letter frequencies (“ETAOIN SHRDLU”)
 - Known plaintext / guess possible words & confirm
 - E.g. “JCKN ECGUCT” =?

Attacks on Caesar Ciphers?

- Brute force: try every possible value of K
 - Work involved?
 - At most 26 “steps”
- Deduction:
 - Analyze letter frequencies (“ETAOIN SHRDLU”)
 - Known plaintext / guess possible words & confirm
 - E.g. “JCKN ECGUCT” =? “HAIL CAESAR”

Attacks on Caesar Ciphers?

- Brute force: try every possible value of K
 - Work involved?
 - At most 26 “steps”
- Deduction:
 - Analyze letter frequencies (“ETAOIN SHRDLU”)
 - Known plaintext / guess possible words & confirm
 - E.g. “JCKN ECGUCT” =? “HAIL CAESAR” $\Rightarrow K=2$
 - Chosen plaintext
 - E.g. Have your spy ensure that the general will send “ALL QUIET”, observe “YJJ OSGCR” $\Rightarrow K=24$

Kerckhoffs' Principle

- Cryptosystems should remain secure even when attacker knows all internal details
 - Don't rely on security-by-obscurity
- Key should be only thing that must stay secret
- It should be easy to change keys
 - Actually distributing these keys is hard, but we will talk about that particular problem later.
 - But key distribution is one of the real...



Better Versions of Rot-K ?

- Consider **$E(M, K) = \text{Rot}\{K_1, K_2, \dots, K_n\}(M)$**
 - i.e., rotate first character by K_1 , second character by K_2 , up through nth character. Then start over with K_1 , ...
 - $K = \{ K_1, K_2, \dots, K_n \}$
- How well do previous attacks work now?
 - Brute force: key space is factor of $26^{(n-1)}$ larger
 - E.g., $n = 7 \Rightarrow 300$ million times as much work
 - Letter frequencies: need more ciphertext to reason about
 - Known/chosen plaintext: works just fine
- Can go further with “chaining”, e.g., 2nd rotation depends on K_2 and first character of ciphertext
 - We just described 2,000 years of cryptography

And Cryptanalysis: ULTRA

Computer Science 161 Fall 2017

Weaver

- During WWII, the Germans used ***enigma***:
 - System was a "rotor machine": A series of rotors, with each rotor permuting the alphabet and every keypress incrementing the settings
 - Key was the selection of rotors, initial settings, and a permutation plugboard
- The British built a system (the "Bombe") to brute-force Enigma
 - Required a known-plaintext (a "crib") to verify decryption: e.g. the weather report
 - Sometimes the brits would deliberately "seed" a naval minefield for a chosen-plaintext attack



One-Time Pad

- Idea #1: use a different key for each message **M**
 - Different = completely independent
 - So: known plaintext, chosen plaintext, etc., don't help attacker
- Idea #2: make the key as long as **M**
- **$E(M,K) = M \oplus K$** ($\oplus = \text{XOR}$)

$$X \oplus 0 = X$$

$$X \oplus X = 0$$

$$X \oplus Y = Y \oplus X$$

$$X \oplus (Y \oplus Z) = (X \oplus Y) \oplus Z$$

\oplus	0	1
0	0	1
1	1	0

One-Time Pad

- Idea #1: use a different key for each message M
 - Different = completely independent
 - So: known plaintext, chosen plaintext, etc., don't help attacker
- Idea #2: make the key as long as M
- $E(M,K) = M \oplus K$ ($\oplus = \text{XOR}$)

$$D(C,K) = C \oplus K$$

$$= M \oplus K \oplus K = M \oplus 0 = M$$

$$X \oplus 0 = X$$

$$X \oplus X = 0$$

$$X \oplus Y = Y \oplus X$$

$$X \oplus (Y \oplus Z) = (X \oplus Y) \oplus Z$$

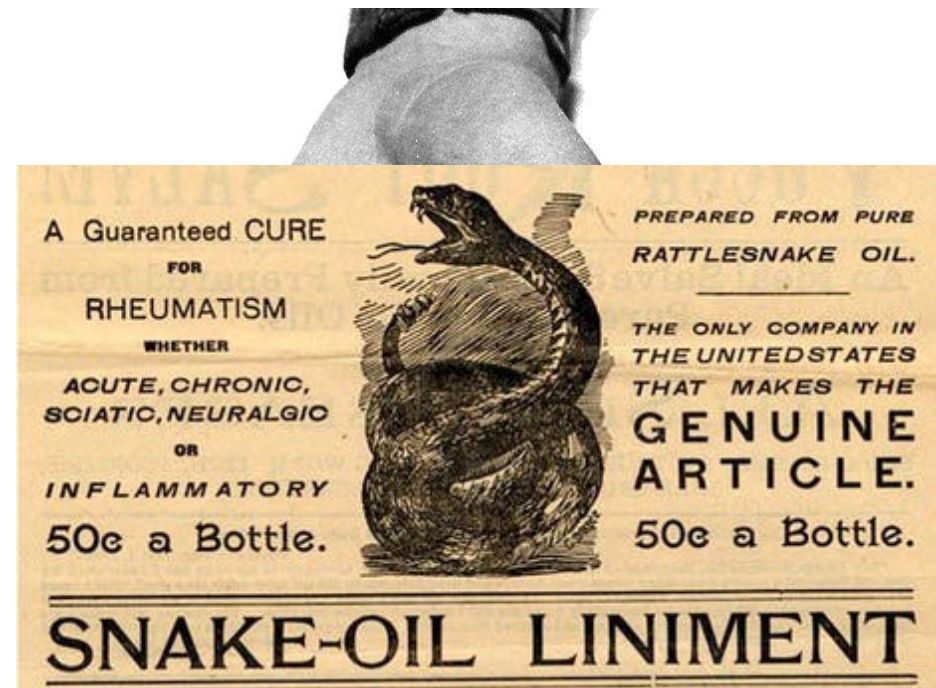
\oplus	0	1
0	0	1
1	1	0

One-Time Pad: Provably Secure!

- Let's assume Eve has partial information about \mathbf{M}
- We want to show: from \mathbf{C} , she does not gain any further information
- Formalization: supposed Alice sends either \mathbf{M}' or \mathbf{M}''
 - Eve doesn't know which; tries to guess based on \mathbf{C}
- Proof:
 - For random, independent \mathbf{K} , all possible bit-patterns for \mathbf{C} are equally likely
 - This holds regardless of whether Alice chose \mathbf{M}' or \mathbf{M}'' , or even if Eve provides \mathbf{M}' and \mathbf{M}'' to Alice and Alice selects which one (IND-CPA)
 - Thus, observing a given \mathbf{C} does not help Eve narrow down the possibilities in any way:

One-Time Pad: Provably Impractical!

- Problem #1: key generation
 - Need truly random, independent keys
- Problem #2: key distribution
 - Need to share keys as long as all possible communication
 - If we have a secure way to establish such keys, just use that for communication in the first place!
 - Only advantage is you can communicate the key in advance: you may have the secure channel now but won't have it later



Two-Time Pad?

- What if we reuse a key **K** jeeeeest once?
- Alice sends **C** = **E(M, K)** and **C'** = **E(M', K)**
- Eve observes **M** \oplus **K** and **M'** \oplus **K**
 - Can she learn anything about M and/or M' ?
- Eve computes **C** \oplus **C'** = (**M** \oplus **K**) \oplus (**M'** \oplus **K**)

Two-Time Pad?

- What if we reuse a key K jeeeeest once?
- Alice sends $\mathbf{C} = \mathbf{E}(\mathbf{M}, \mathbf{K})$ and $\mathbf{C}' = \mathbf{E}(\mathbf{M}', \mathbf{K})$
- Eve observes $\mathbf{M} \oplus \mathbf{K}$ and $\mathbf{M}' \oplus \mathbf{K}$
 - Can she learn anything about \mathbf{M} and/or \mathbf{M}' ?
- Eve computes $\mathbf{C} \oplus \mathbf{C}' = (\mathbf{M} \oplus \mathbf{K}) \oplus (\mathbf{M}' \oplus \mathbf{K})$
 - $= (\mathbf{M} \oplus \mathbf{M}') \oplus (\mathbf{K} \oplus \mathbf{K})$
 - $= (\mathbf{M} \oplus \mathbf{M}') \oplus \mathbf{0}$
 - $= \mathbf{M} \oplus \mathbf{M}'$
- Now she knows which bits in \mathbf{M} match bits in \mathbf{M}'
- And if Eve already knew \mathbf{M} , now she knows \mathbf{M}' !
 - Even if not, Eve can guess \mathbf{M} and ensure that \mathbf{M}' is consistent



VENONA:

Pad Reuse in the Real World

Computer Science 161 Fall 2017

Weaver

- The Soviets used one-time pads for communication from their spies in the US
 - After all, it is provably secure!
- During WWII, the Soviets started reusing key material
 - Uncertain whether it was just the cost of generating pads or what...
- VENONA was a US cryptanalysis project designed to break these messages
 - Included confirming/identifying the spies targeting the US Manhattan project
 - Project continued until 1980!
- ***Not declassified until 1995!***
 - So secret even President Truman wasn't informed about it.
 - But the Soviets found out about it in 1949, but their one-time pad reuse was fixed after 1948 anyway



Modern Encryption:

Block cipher

- A function $\mathbf{E} : \{0, 1\}^b \times \{0, 1\}^k \rightarrow \{0, 1\}^b$. Once we fix the key \mathbf{K} (of size k bits), we get:
- $\mathbf{EK} : \{0, 1\}^b \rightarrow \{0, 1\}^b$ denoted by $\mathbf{E_K(M)} = \mathbf{E(M, K)}$.
 - (and also $\mathbf{D(C, K)}$, $\mathbf{E(M, K)}$'s inverse)
- Three properties:
 - Correctness:
 - $\mathbf{E_K(M)}$ is a permutation (bijective function) on b -bit strings
 - Bijective \Rightarrow invertible
 - Efficiency: computable in μsec 's
 - Security:
 - For unknown \mathbf{K} , “behaves” like a random permutation
- Provides a building block for more extensive encryption

DES (Data Encryption Standard)

- Designed in late 1970s
- Block size 64 bits, key size 56 bits
- NSA influenced two facets of its design
 - Altered some subtle internal workings in a mysterious way
 - Reduced key size 64 bits \Rightarrow 56 bits
 - Made brute-forcing feasible for attacker with massive (for the time) computational resources
- Remains essentially unbroken 40 years later!
 - The NSA's tweaking hardened it against an attack "invented" a decade later
- However, modern computer speeds make it completely unsafe due to small key size

Today's Go-To Block Cipher: AES (Advanced Encryption Standard)

- 20 years old, standardized 15 years ago...
- Block size 128 bits
- Key can be 128, 192, or 256 bits
 - 128 remains quite safe; sometimes termed “AES-128”
- As usual, includes encryptor and (closely-related) decryptor
 - How it works is beyond scope of this class
- Not proven secure
 - But no known flaws
 - The NSA uses it for Top Secret communication with 256b keys:
stuff they want to be secure **for 40 years including possibly unknown breakthroughs!**
 - so we assume it is a secure block cipher

How Hard Is It To Brute-Force 128-bit Key?

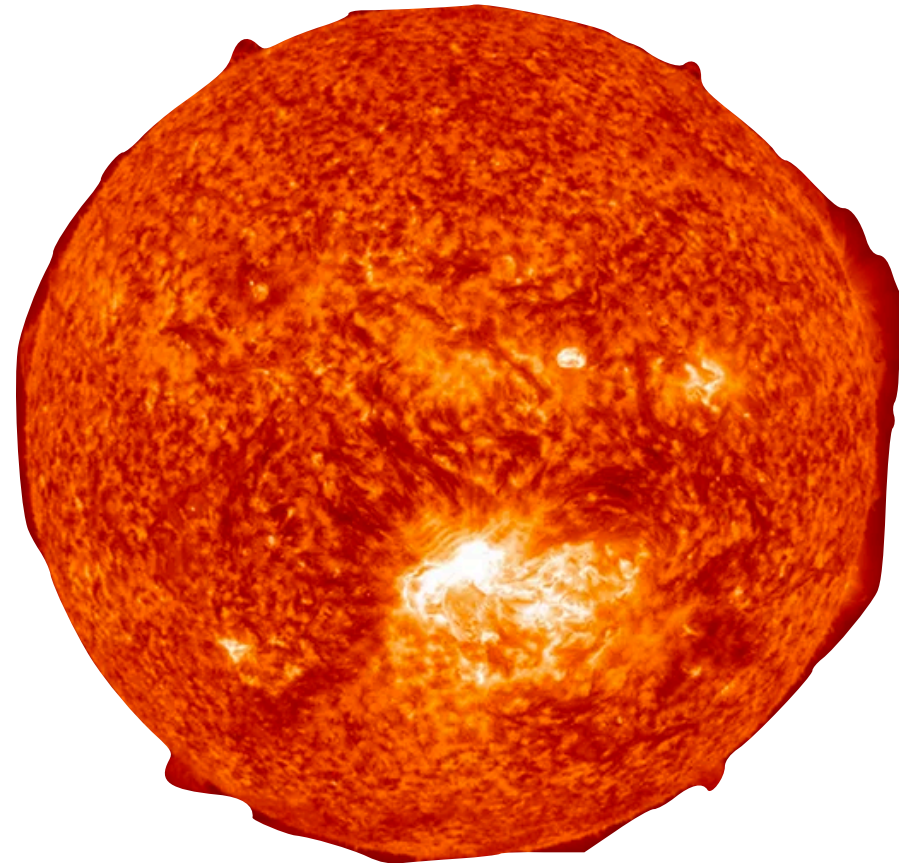
- 2^{128} possibilities – well, how many is that?
- Handy approximation: $2^{10} \approx 10^3$
- $2^{128} = 2^{10 \cdot 12.8} \approx (10^3)^{12.8} \lesssim (10^3)^{13} \approx 10^{39}$

How Hard Is It To Brute-Force 128-bit Key?

- 2^{128} possibilities – well, how many is that?
- Handy approximation: $2^{10} \approx 10^3$
- $2^{128} = 2^{10 \cdot 12.8} \approx (10^3)^{12.8} \approx (10^3)^{13} \approx 10^{39}$
- Say we build massive hardware that can try 10^9 (1 billion) keys in 1 nanosecond (a billionth of a second)
 - So 10^{18} keys/sec
 - Thus, we'll need $\approx 10^{21}$ sec
- How long is that?
 - One year $\approx 3 \times 10^7$ sec
 - So need $\approx 3 \times 10^{13}$ years ≈ 30 trillion years

What about a 256b key in a year?

- Time to start thinking in ***astronomical*** numbers:
 - If each brute force device is 1mm^3 ...
 - We will need 10^{52} of these things...
- 10^{43} cubic meters...
- Or the volume of ***7×10^{15} suns!***
- Brute force is ***not a factor*** against modern block ciphers...
IF the key is actually random!



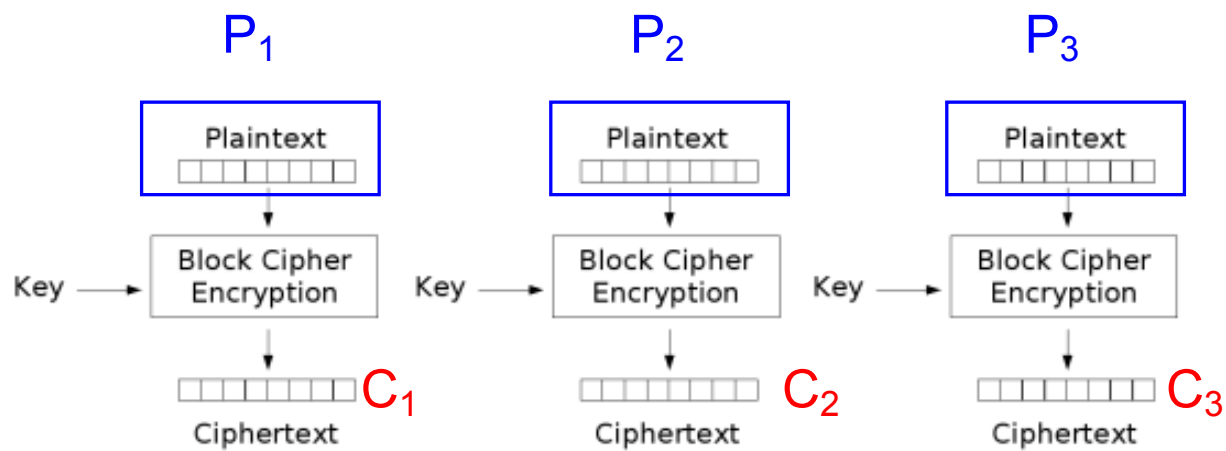
Issues When Using the Building Block

- Block ciphers can only encrypt messages of a certain size
 - If **M** is smaller, easy, just pad it (details omitted)
 - If **M** is larger, can repeatedly apply block cipher
 - Particular method = a “block cipher mode”
 - Tricky to get this right!
- If same data is encrypted twice, attacker knows it is the same
 - Solution: incorporate a varying, known quantity (IV = “initialization vector”)

Electronic Code Book (ECB) mode

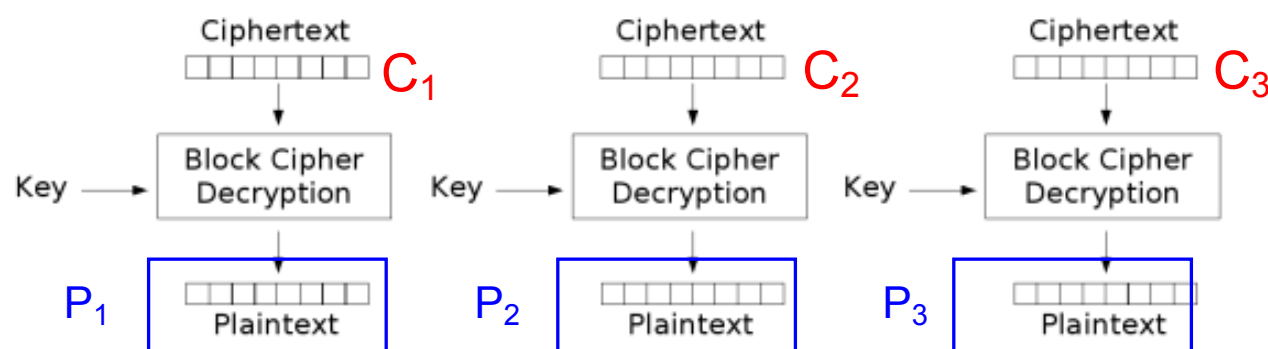
- Simplest block cipher mode
- Split message into b-bit blocks $\mathbf{P}_1, \mathbf{P}_2, \dots$
- Each block is enciphered independently, separate from the other blocks
 $\mathbf{C}_i = \mathbf{E}(\mathbf{P}_i, \mathbf{K})$
- Since key \mathbf{K} is fixed, each block is subject to the same permutation
- (As though we had a “code book” to map each possible input value to its designated output)

ECB Encryption



Electronic Codebook (ECB) mode encryption

ECB Decryption



Electronic Codebook (ECB) mode decryption

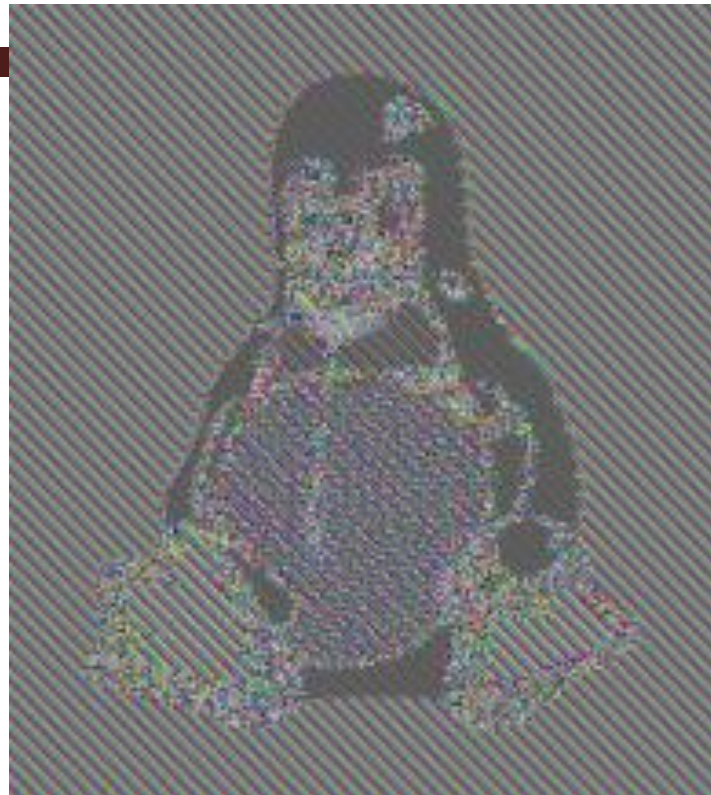
Problem: Relationships between P_i 's reflected in C_i 's

IND-CPA and ECB?

- Of course not!
- **M, M'** is 2x the block length...
 - **M** = all 0s
 - **M'** = 0s for 1 block, 1s for the 2nd block
- This has catastrophic implications in the real world...



Original image, RGB values split into a bunch of b-bit blocks



Encrypted with ECB and interpreting ciphertext directly as RGB



Later (identical) message again encrypted with ECB

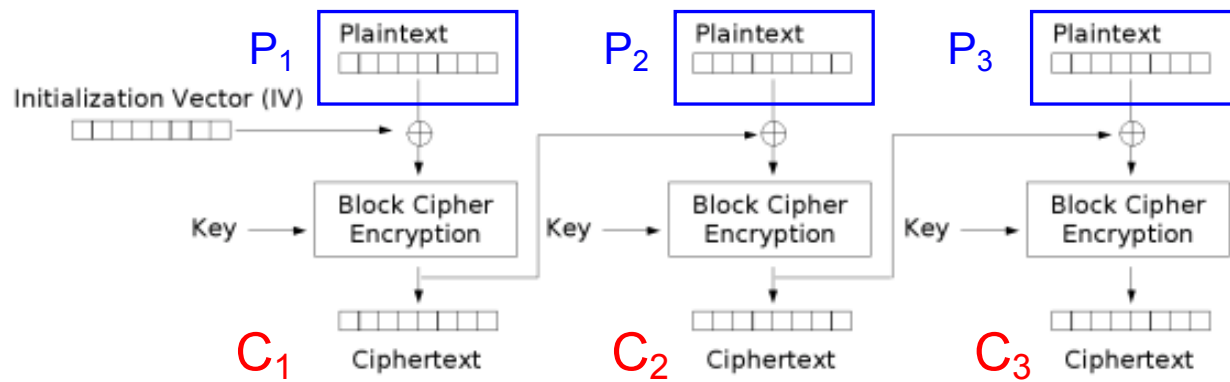
Building a Better Cipher Block Mode

- Ensure blocks incorporate more than just the plaintext to mask relationships between blocks. Done carefully, either of these works:
 - Idea #1: include elements of prior computation
 - Idea #2: include positional information
- Plus: need some initial randomness
 - Prevent encryption scheme from determinism revealing relationships between messages
 - Introduce initialization vector (IV):
 - IV is a public **nonce**, a use-once unique value: Easiest way to get one is generate it randomly
- Example: Cipher Block Chaining (CBC)

CBC Encryption

$E(\text{Plaintext}, K)$:

- If b is the block size of the block cipher, split the plaintext in blocks of size b : P_1, P_2, P_3, \dots
- Choose a random IV (do not reuse for other **messages**)
- Now compute:



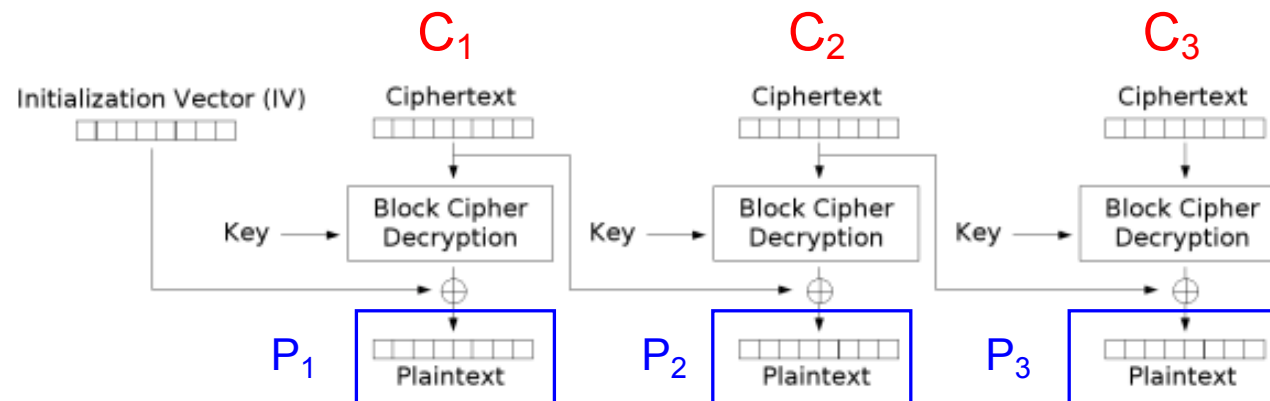
Cipher Block Chaining (CBC) mode encryption

- Final ciphertext is (IV, C_1, C_2, C_3) . This is what Eve sees.

CBC Decryption

$D(\text{Ciphertext}, K)$:

- Take **IV** out of the ciphertext
- If **b** is the block size of the block cipher, split the ciphertext in blocks of size **b**: C_1, C_2, C_3, \dots
- Now compute this:

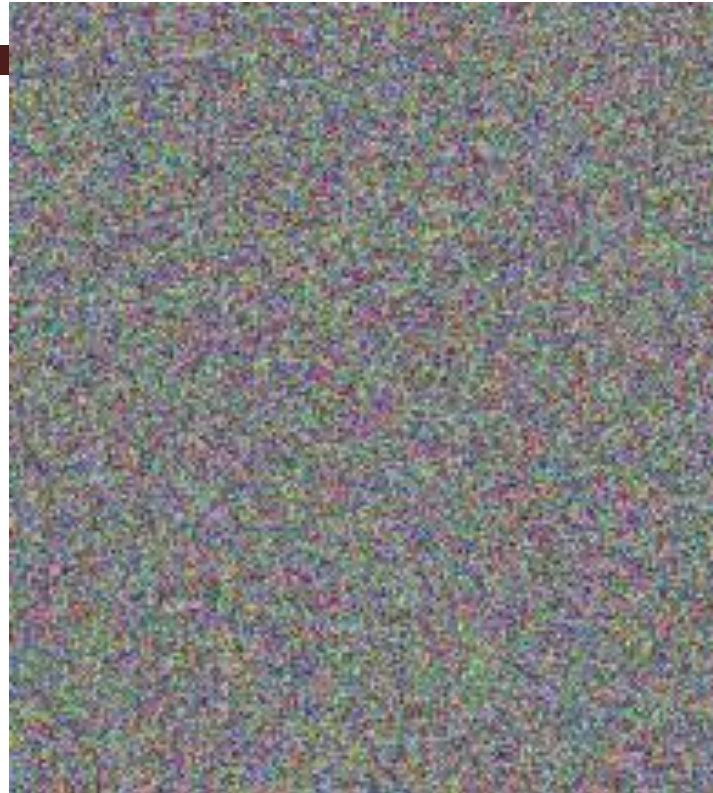


Cipher Block Chaining (CBC) mode decryption

- Output the plaintext as the concatenation of P_1, P_2, P_3, \dots



Original image, RGB values split into a bunch of b-bit blocks



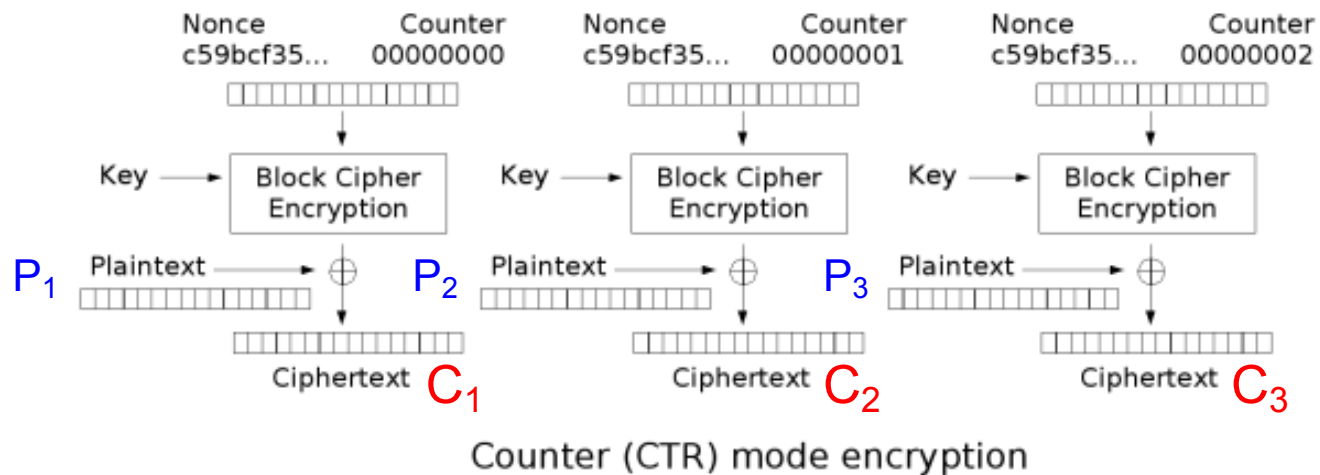
Encrypted with CBC

CBC Mode...

- Widely used
- Issue: sequential encryption, can't parallelize encryption
 - **Must** finish encrypting block b before starting $b+1$
 - But you can parallelize decryption
- Parallelizable alternative: CTR (Counter) mode
- Security: If no reuse of nonce, both are provably secure (IND-CPA) assuming the underlying block cipher is secure

CTR Mode Encryption

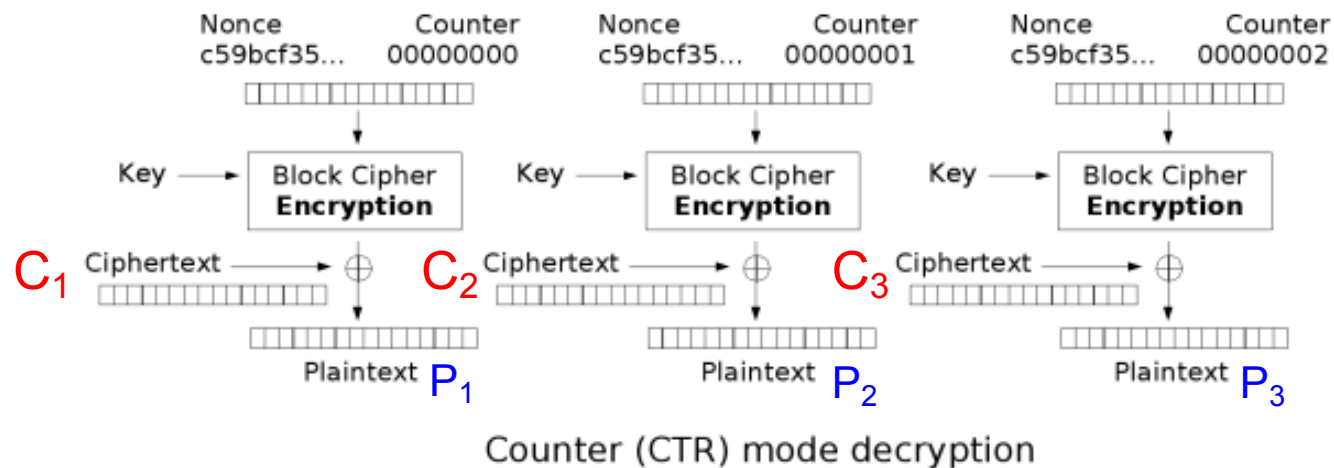
(Nonce = Same as IV)



Important that **nonce/IV** does not repeat across different encryptions.

Choose at random!

Counter Mode Decryption



Note, CTR decryption uses block cipher's *encryption*, **not** decryption

Thoughts on CTR mode...

- CTR mode is actually a stream cipher (more on those later):
 - You no longer need to worry about padding which is nice
- CTR mode is fully parallelizeable for encryption as well as decryption

NEVER EVER EVER use CTR Mode!

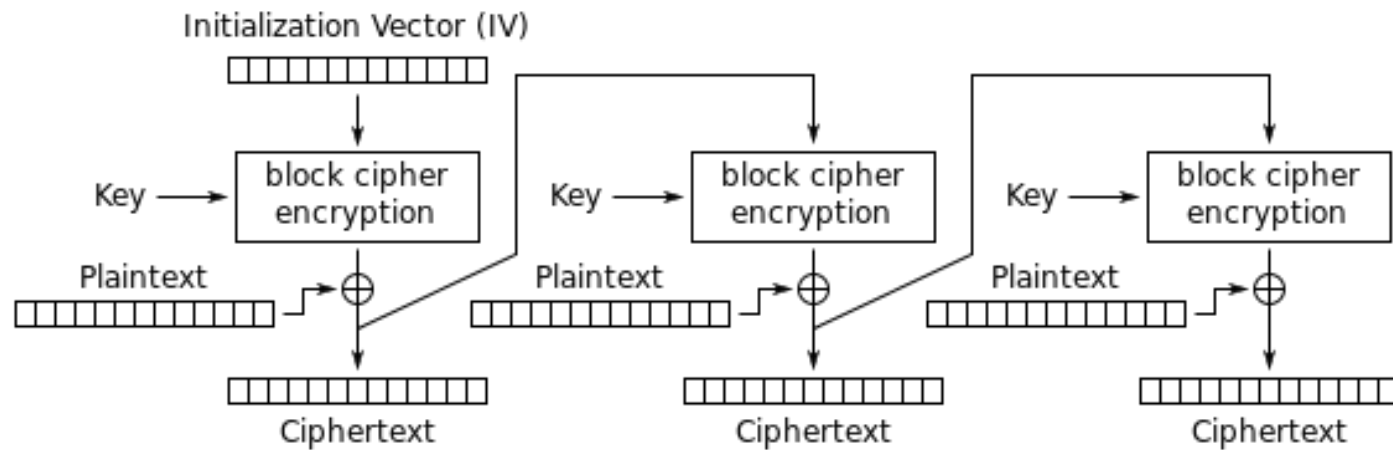
- What happens if you reuse the IV in CBC...
 - Its bad but not catastrophic:
you fail IND-CPA but the damage may be tolerable:
 - $M = \{A, A, B\}$
 $M' = \{A, B, B\}$
Adversary can see that the first part of M and M' are the same, but not the later part
- What happens if you reuse the IV in CTR mode?
 - It is **exactly** like reusing a one-time pad!
- An example of a system which fails badly...
 - CTR mode is **theoretically** as secure as CBC when used properly...
 - But when it is misused it fails catastrophically:
Personal bias: I believe we need systems that are still robust **when implemented incorrectly**



What To Use Then?

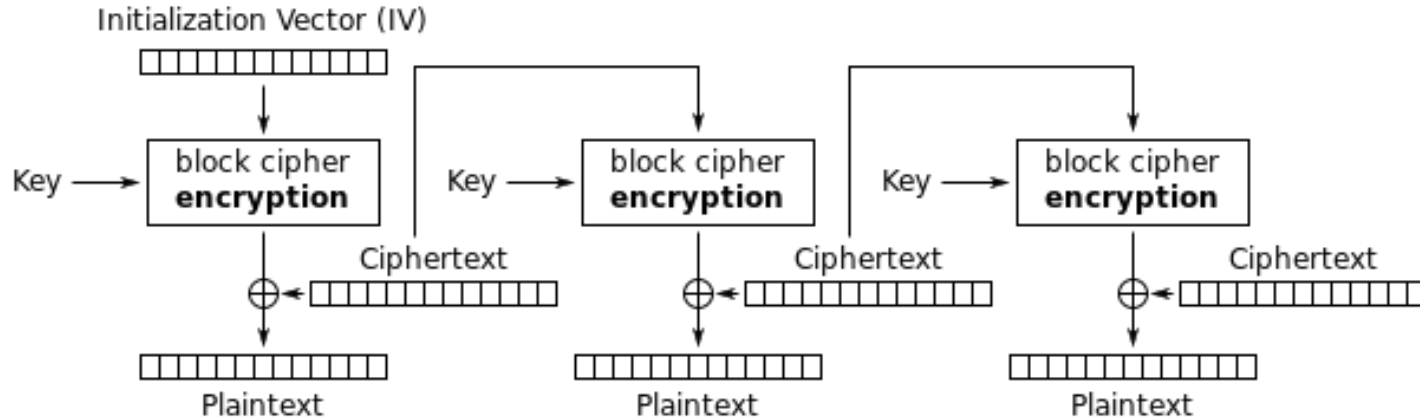
- What if you want a cipher mode where you don't need to pad (like CTR mode)?
 - But you want the robust to screwup properties of CBC mode?
- Idea: lets do it CTR-like (xor plaintext with block cipher output), but...
- Instead of the next block input being an incremented counter...
have the next block be the previous ciphertext
- Still lacks integrity however, we'll fix that next time...

CFB Encryption



Cipher Feedback (CFB) mode encryption

CFB Decryption



Cipher Feedback (CFB) mode decryption

CFB doesn't need to pad...

- Since the encryption is XORed with the plaintext...
 - You can end on a "short" block without a problem
 - So more convenient than CBC mode
- But similar security properties as CBC mode
 - Sequential encryption, parallel decryption
 - Same error propagation effects
 - Effectively the same for IND-CPA
- But a bit worse if you reuse the IV