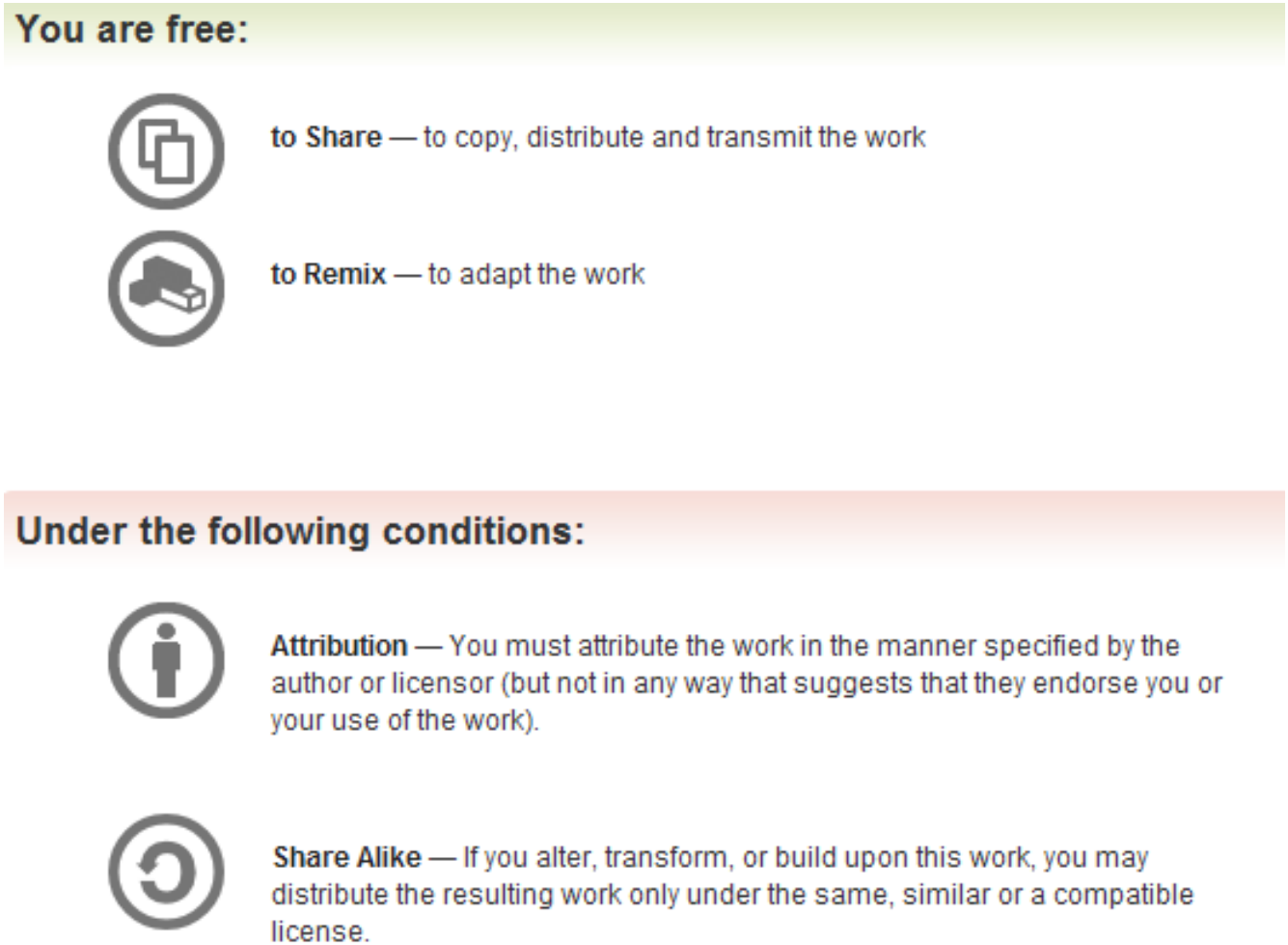
Understanding Cryptology

Core Concepts

Dr. Kerry A. McKay

All materials is licensed under a Creative Commons “Share Alike” license.

- http://creativecommons.org/licenses/by-sa/3.0/



Goals

- Provide an accessible introduction to cryptology
  - No math degree required
- Provide abstract ideas
  - Understand form of many algorithms rather than details of just a few
- Cover a little of everything
  - Lay groundwork for deeper study in specialized areas

Day 1

- Overview of cryptology
- Introduction to cryptography
- A high level view of cryptography and cryptanalysis
- Symmetric cryptography
- Asymmetric cryptography
- Additional primitives

Day 2

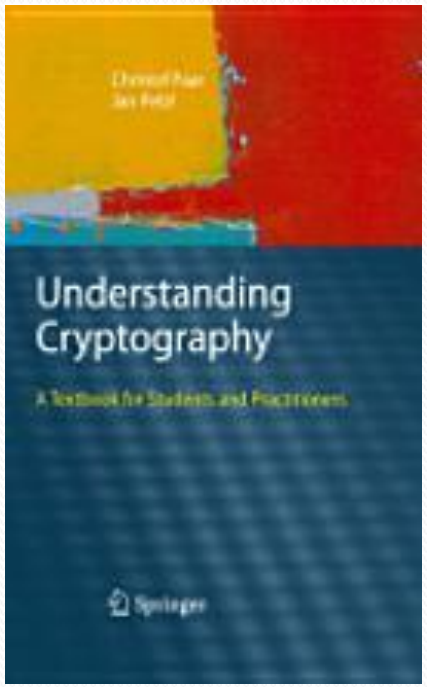
- Protocols
- Cryptanalysis
- Case studies
- Standardization
- Summary and lessons

Administrivia

- Sign in
- There is some math in here, but I've tried to keep it to a minimum
- This course was designed with a bachelor's level computer scientist in mind
  - If you don't understand something, just ask! ☺

Recommended reading

- Book for course
  - Understanding Cryptography: A Textbook for Students and Practitioners
    - Paar and Pelzl



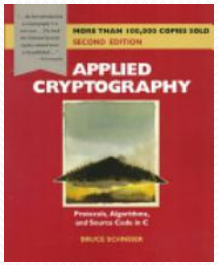
- Companion website
  - http://www.crypto-textbook.com/
  - Slides developed by authors
  - Videos

Additional Suggestions

Applied

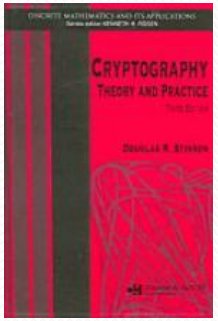
- Cryptography Engineering
  - Ferguson, Schneier, Kohno

- Applied Cryptography
  - Schneier

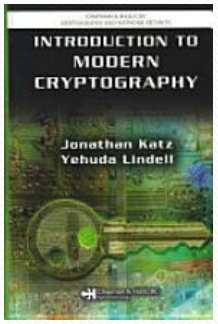


Theory

- Cryptography: Theory and Practice
  - Stinson



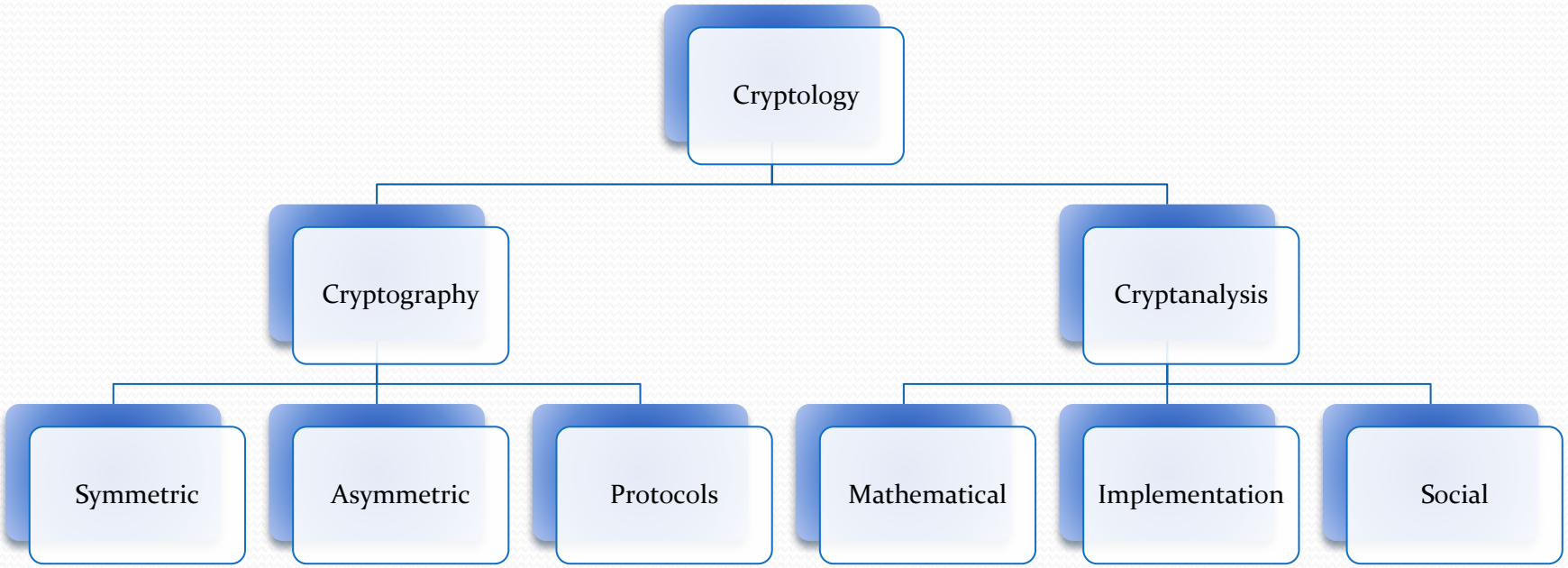
- Introduction to Modern Cryptography
  - Katz, Lindell



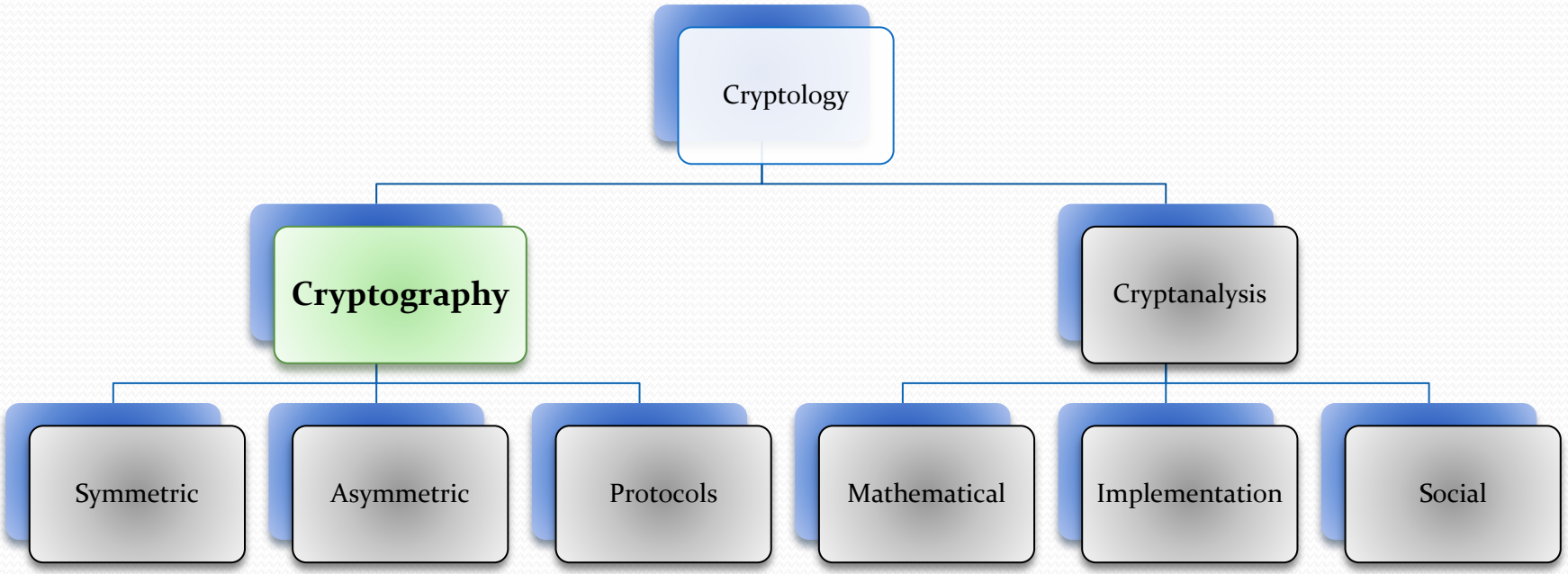
What is Cryptology?

- **Cryptology** is "the scientific study of cryptography and cryptanalysis" (according to Merriam-Webster)
  - People often say "cryptography" to mean both
- **Cryptography** is the science of secret writing with the goal of hiding the meaning of a message
  - This is different from steganography, where the presence of the message is hidden
- **Cryptanalysis** is the science and art of *breaking cryptosystems*

Crypto topics covered in this course



Cryptography



Cryptography

- Cryptography has expanded over the years
- Symmetric ciphers
  - Both encryption and decryption use the same key
- Asymmetric ciphers
  - Different keys for encryption and decryption
- Protocols
  - Inherently cryptographic
  - Applied cryptography

- We'll talk about each of these more later

What can cryptography do for you?

- People sometimes think that adding cryptography makes something secure
  - Not true
- Cryptography can't solve everything
  - Buffer overflow, social engineering, malware, etc.
- For the problems it can solve, it needs to be used correctly

Cryptography in information assurance

- Confidentiality
  - Prevents unauthorized parties from accessing information
    - Encryption removes meaning from information
- Integrity
  - Ensures that data cannot be modified without detection
    - Hash functions make it extremely difficult to change information
- Authentication
  - Ensure that a message was created by a particular party
    - Key only known by party

Start with a classic

- Let's begin with a concrete example
- Caesar (shift) cipher

- To do this, we need to take a quick look at modular addition

Modular Arithmetic

- What time is it?
- What time will it be in 24 hours?

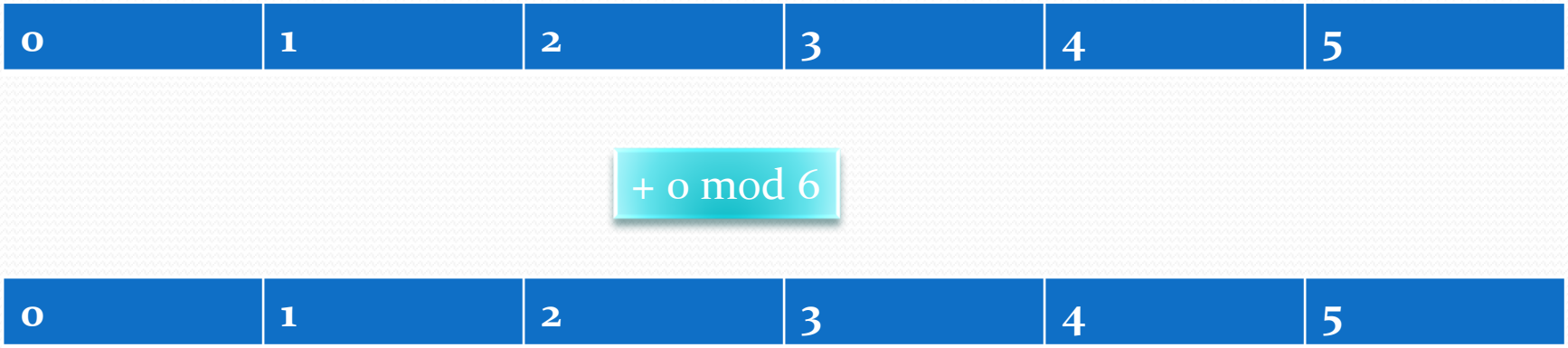
- Modular arithmetic deals with operations in finite sets
  - Hours are mod 24 for military time
    - {0,1,2,…,23}
  - Hours are mod 12 on standard clocks
    - {12, 1, 2, …, 11}
    - The zero is written as 12 (12 mod 12 = 0)
  - Minutes are mod 60
    - {0, 1, 2, …, 59}
  - Days, weeks, months etc. can also be expressed this way

Modular Arithmetic

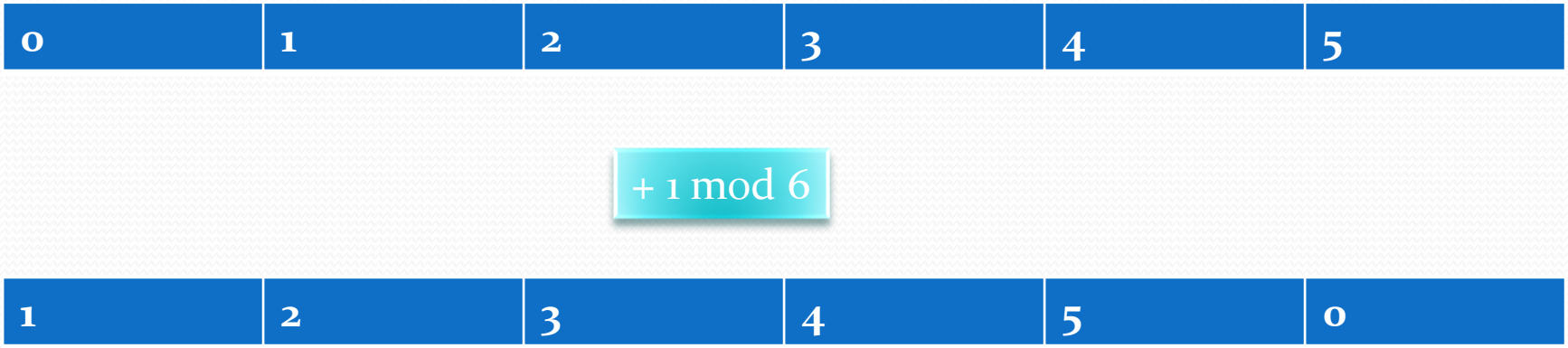
- To calculate A mod B, take the remainder
  - $A = qB + r$, where $0 \leq r < B$
    - By the division theorem
    - r is the remainder of A/B
  - A mod B = r = A - qB

- Examples
  - 12+12 mod 24 = 0
  - 12 + 12 mod 20 = 4
  - 5*1 mod 8 = 5
  - 5*5 mod 20 = 5

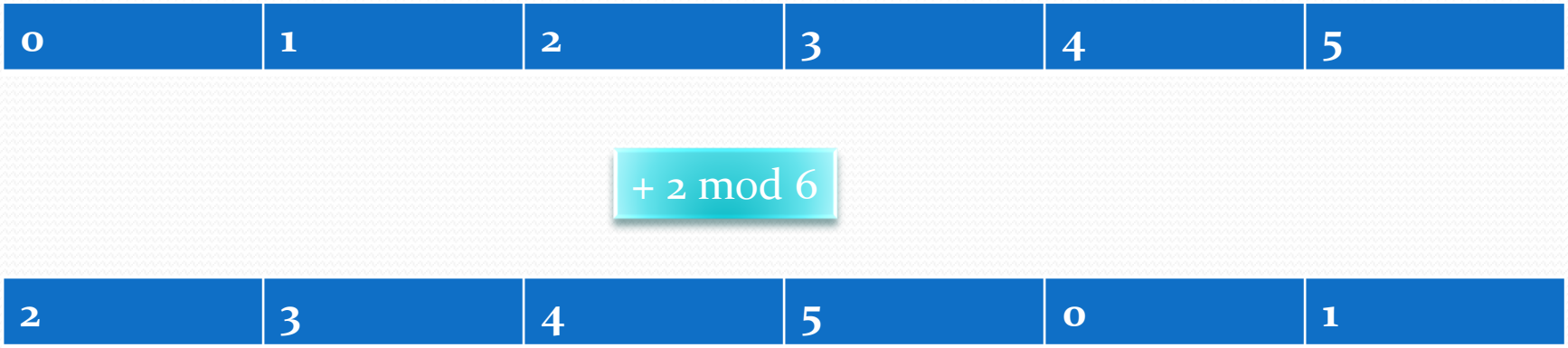
Example



Example



Example



Back to Caesar's cipher

- Map each letter, A-Z, to an integer, 0-25
- Select letter as secret key
- Convert message and key to integers
- Add key to each character modulo 26
- Convert back to letters

Letter	#	Letter	#
A	0	N	13
B	1	O	14
C	2	P	15
D	3	Q	16
E	4	R	17
F	5	S	18
G	6	T	19
H	7	U	20
I	8	V	21
J	9	W	22
K	10	X	23
L	11	Y	24
M	12	Z	25

Example

- Message: HELLO WORLD
  - 7 4 11 11 14 22 14 17 11 3
- Key: E
  - 4
- Encryption
  - 11 8 15 15 18 0 18 21 15 7
  - LIPPSASVPH

Letter	#	Letter	#
A	0	N	13
B	1	O	14
C	2	P	15
D	3	Q	16
E	4	R	17
F	5	S	18
G	6	T	19
H	7	U	20
I	8	V	21
J	9	W	22
K	10	X	23
L	11	Y	24
M	12	Z	25

Example

- Now let's decrypt it
- Inverse of addition mod 26 is subtraction mod 26
- Subtract 4 from each integer

- LIPPSASVPH
  - 11 8 15 15 18 0 18 21 15 7

- Decryption
  - 7 4 11 11 14 22 14 17 11 3
  - HELLOWORLD

Letter	#	Letter	#
A	0	N	13
B	1	O	14
C	2	P	15
D	3	Q	16
E	4	R	17
F	5	S	18
G	6	T	19
H	7	U	20
I	8	V	21
J	9	W	22
K	10	X	23
L	11	Y	24
M	12	Z	25

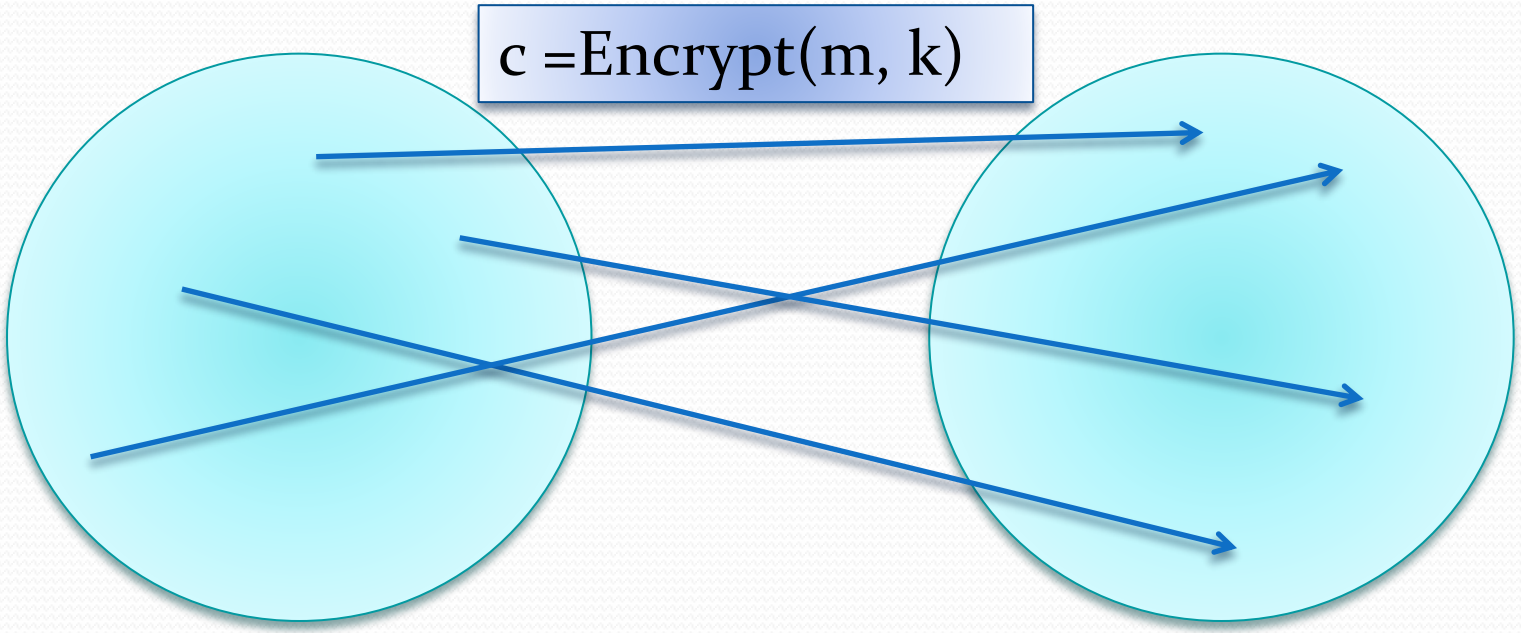
What's really going on?

- Encryption and decryption are abstract concepts

- Let's abstract this further
- Plaintext: unencrypted message
- Ciphertext: encrypted message

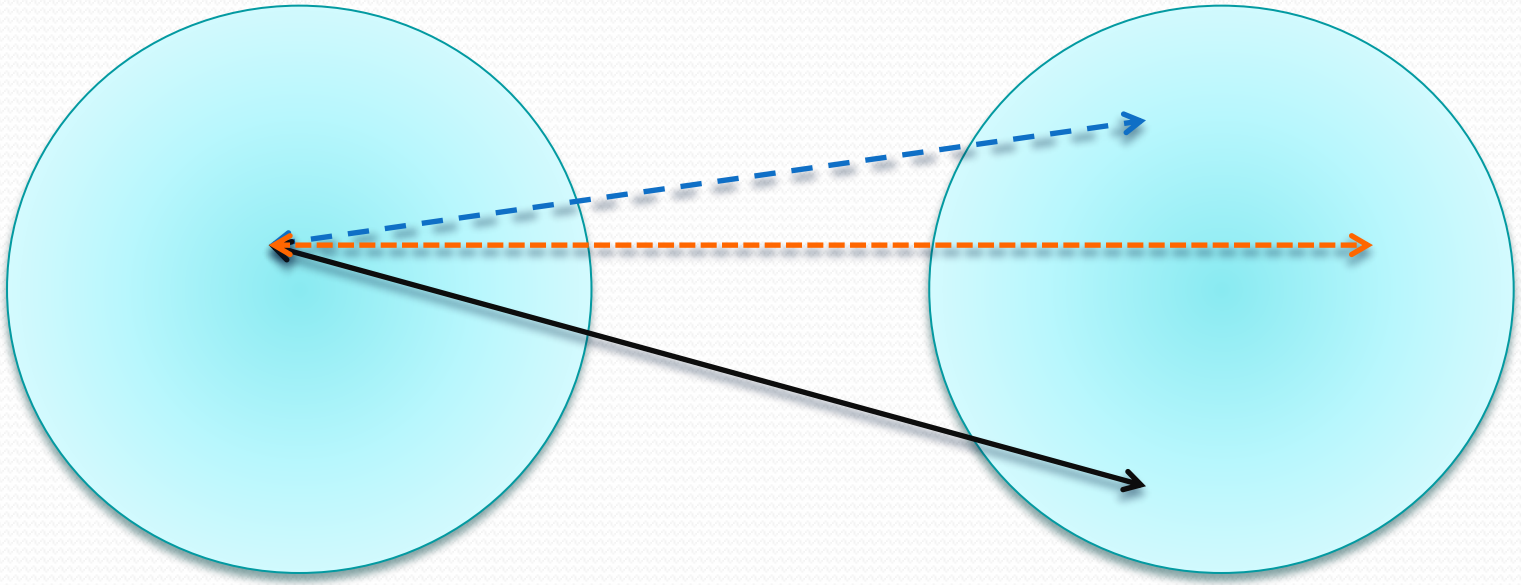
The basic picture

- The set of all possible plaintexts
- The set of all possible ciphertexts



The key

- The key determines the map destination
  - In this picture, three different keys cause the same element to map to three different values



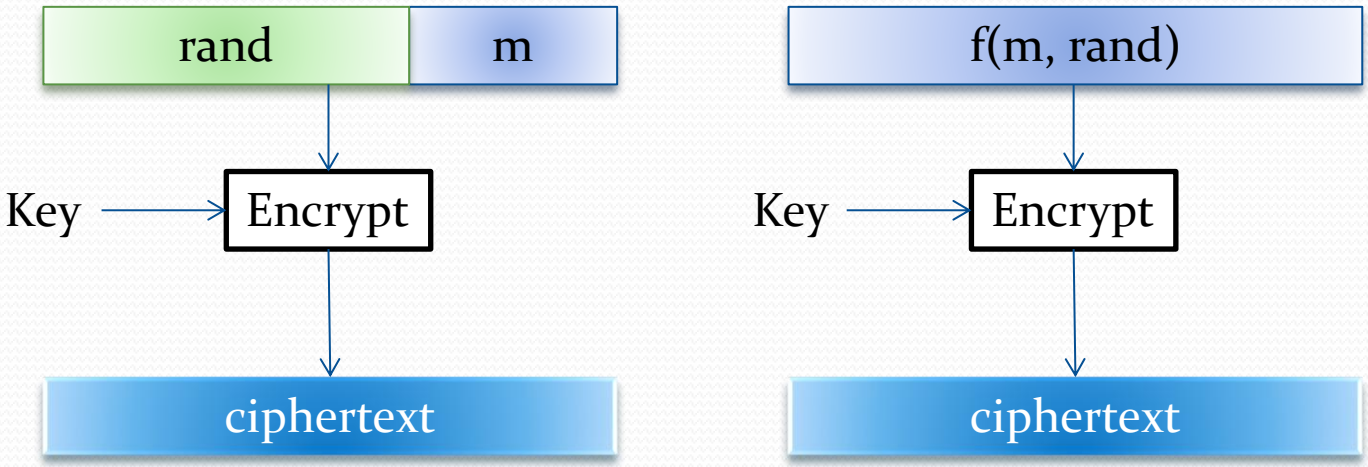
Is the mapping one-to-one?

- Sometimes the encryption is deterministic
  - Every time you run it with the same plaintext-key pair, you get the same result
- Sometimes the encryption is probabilistic
  - You may get different results
  - The same key with the same message will give one of a set of results

- Decryption is always deterministic
  - You MUST get the original message back

Probabilistic mappings

- Sometimes randomness is part of the encryption process

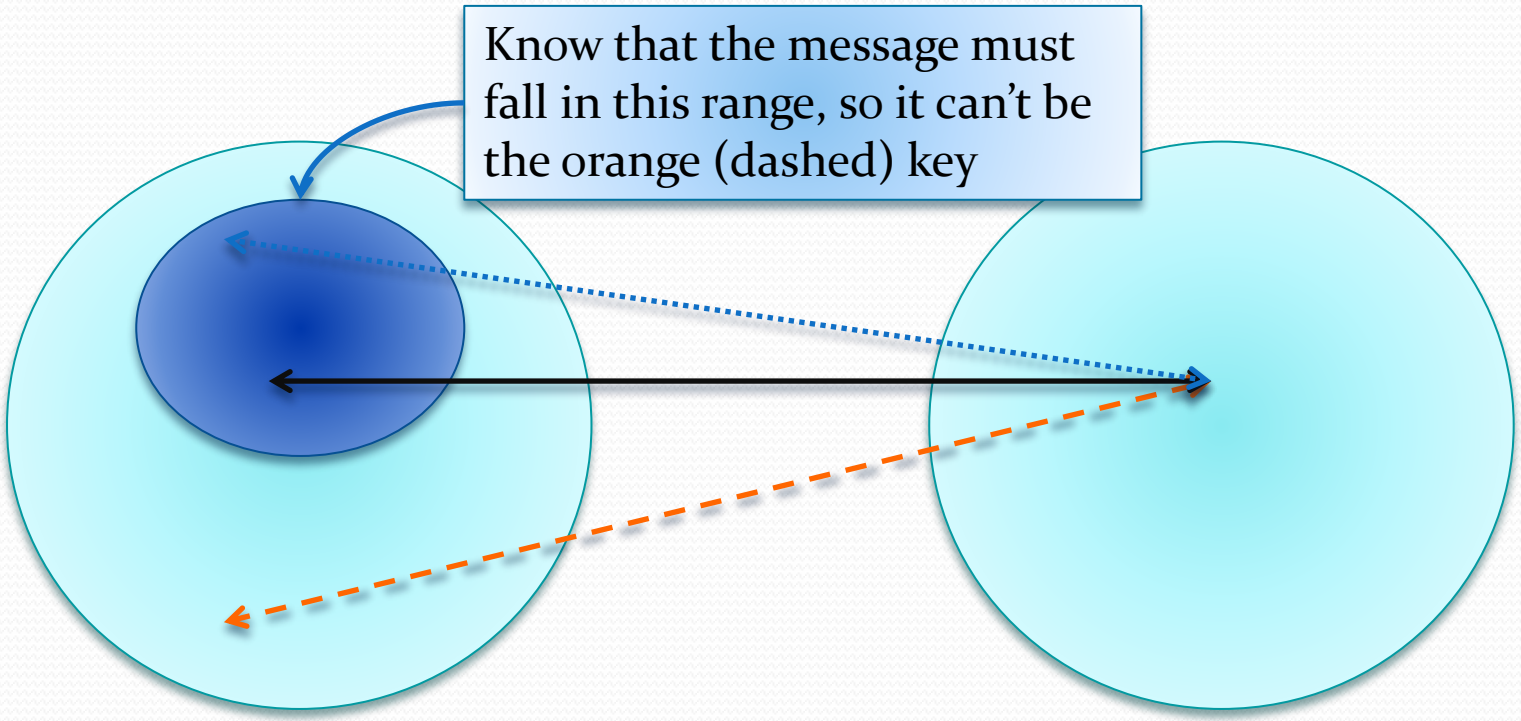


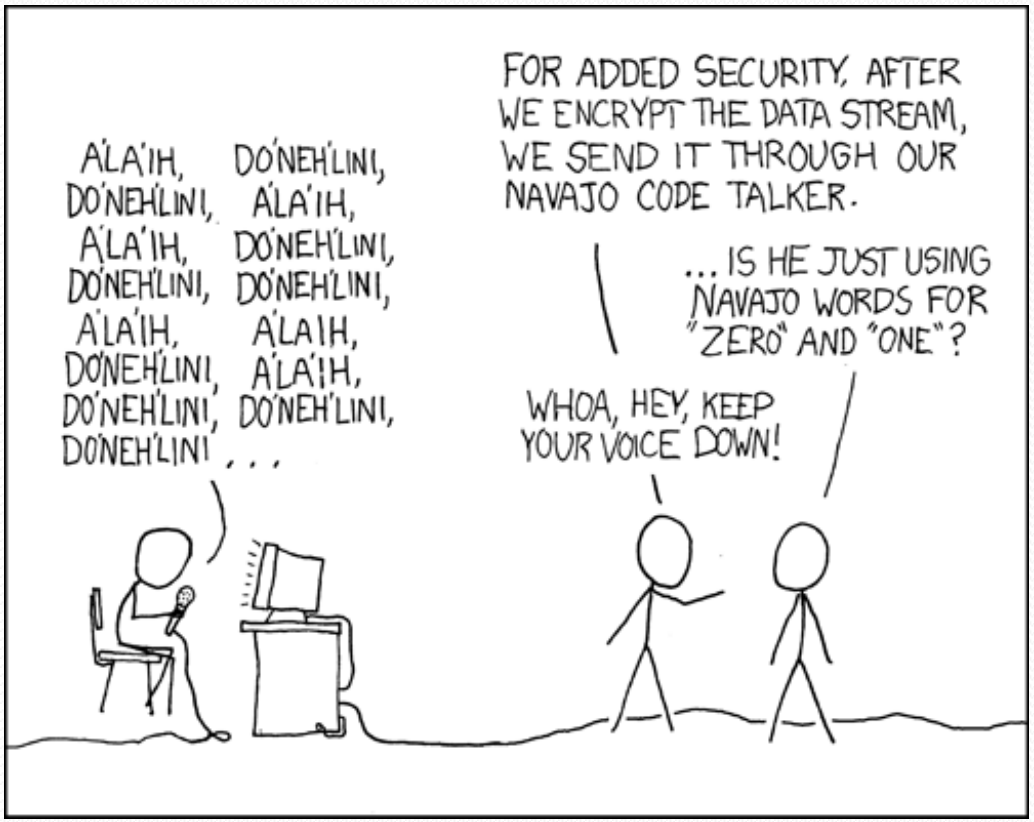
Enter the adversary

- Throughout this course we will refer to the party attacking the system as the adversary
- The adversary's goal is to find the plaintext or key with much less effort than guessing
  - The key is better, because then they can easily find *all* plaintexts that were encrypted under the same key

The adversary's view

- The adversary knows the ciphertext (and maybe more)
  - Reduce the work any way they can





http://xkcd.com/257/

The role of Cryptanalysis in Cryptography

- Cryptanalysis plays a very important role in designing cryptosystems
  - State-of-the-art analysis techniques drive the designs
  - If an attack exists, a new algorithm should be resilient to it
  - Determines the security of an algorithm
- If an algorithm has a 100-bit key but there is an attack that requires only 2^{40} encryptions, then the algorithm only provides 40 bits of security
  - Key length determines the work for brute force, but not for smarter attacks

Kerckhoffs' Principle

- *A cryptosystem should be secure even if the attacker knows all details about the system, with the exception of the secret key. In particular, the system should be secure when the attacker knows the encryption and decryption algorithms.*

- It all comes down to the key
- Security through obscurity is *not* a good idea

Security in Cryptography

- An algorithm is only as secure as the most advanced cryptanalysis against it
- There is no silver bullet, there is no spoon, and there is no one definition of “secure”
- In fact, our definition of secure changes every day

- A better question might be “What does it take to be considered secure today?”

Key length

- We just saw that cryptography provides a map from one set to another
  - The only way to get the original message back from the ciphertext is to invert the map (go backwards)
  - This means choosing the right key
- For an algorithm to be considered secure, it needs to be *computationally infeasible* for an adversary to get the key with effort under a threshold
  - The key needs to come from a large enough key space (set of possible keys)
  - That threshold keeps rising

Key length and key space

- Key length is measured in bits
- A key that is *n* bits long provides a key space of size 2^n
- Assuming
  - All keys are possible
  - No two keys produce the same mapping

- How big does *n* need to be?

Key length then and now

- In 1976, DES was the standard algorithm, with a 56-bit key
- In 1980, Skipjack had an 80-bit key
- TDEA (triple DES) has 112 or 168 bits of key, depending on version
  - 56-bit version for backward compatibility
- In 1998, NIST called for a new standard with key sizes 128, 192, and 256

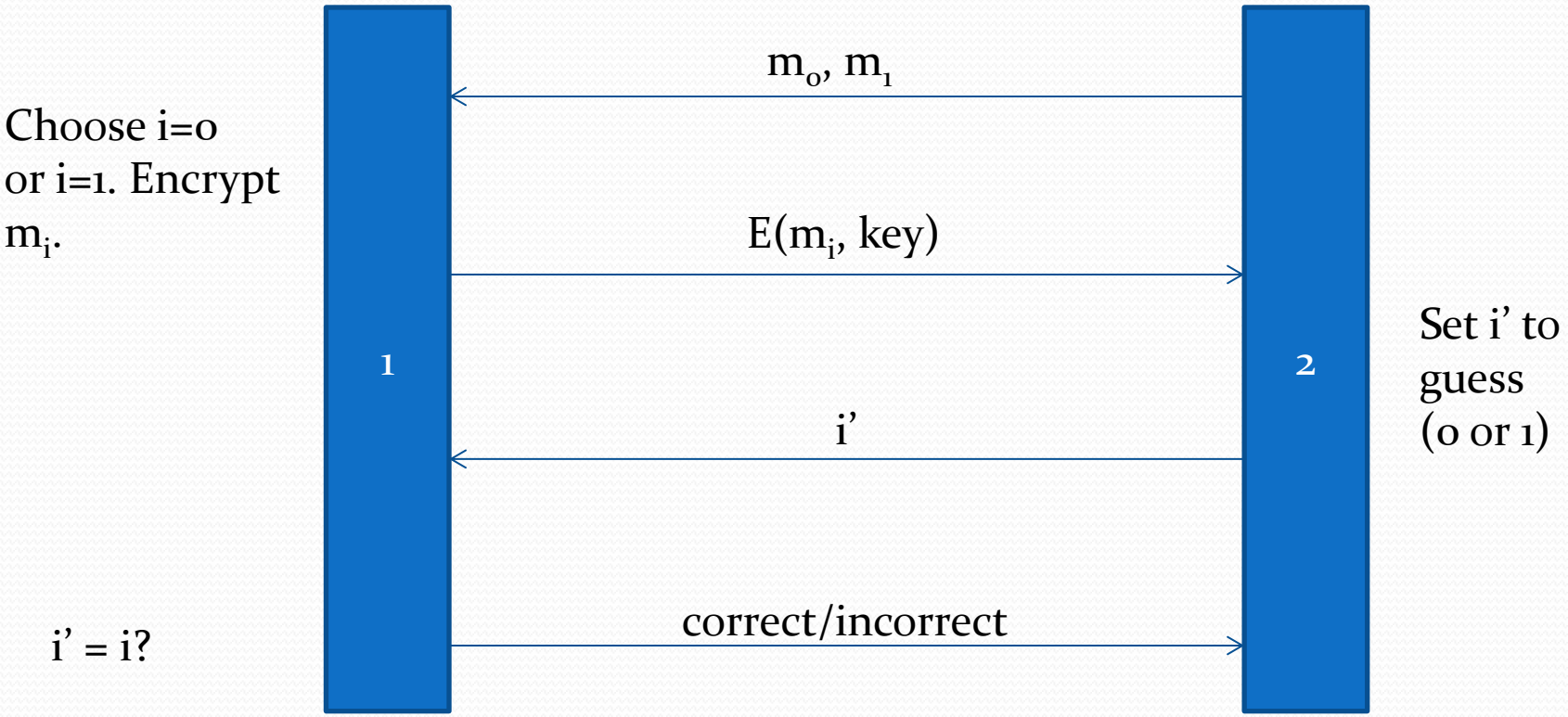
- Today, you should have at least 112 bits of security
  - 80 bits is still acceptable, but is being phased out
  - After 2013, the minimum will be 112

- (Note: these sizes are for symmetric algorithms. Asymmetric algorithms, in general, require larger keys)

A game: adversarial indistinguishability

- One of the many definitions of security involves a game
  - Two players
- Rules
  - Player 2 chooses two messages and tells player 1
  - Player 1 chooses one of the two messages
    - Encrypts the message
    - Sends the encrypted message back to player 2
  - Player 2's task is to determine whether the encrypted message is an encryption of m_0 or m_1

Adversarial indistinguishability (continued)



Exercise

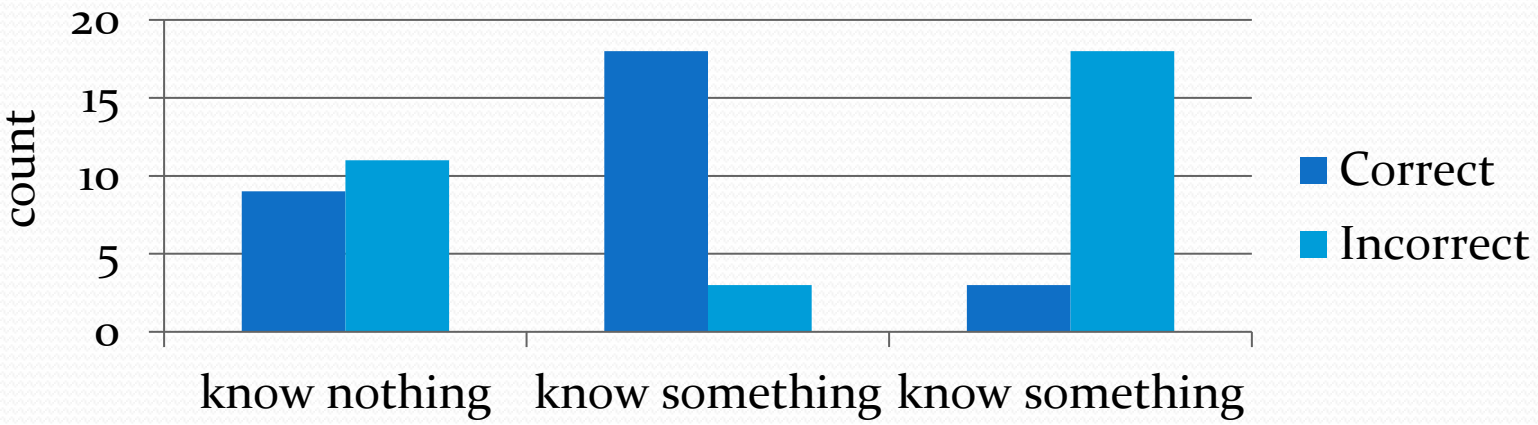
- Pair up and decide who is player 1 and who is player two
- Player 1
  - Come up with a **simple deterministic** function to encrypt a number from 1 to 100 (remember it needs to be invertible)
  - For example, f(x) = 2x
  - You need to be able to compute this in your head!
- Player 2
  - Choose two numbers from 1 to 100 (but not the same one twice!) to use as messages
- Play a few rounds
  - Keep track of when player 2 is right and wrong
  - player 1 has to use the same function every time

Exercise (continued)

- After a few rounds,
  - Did you begin to see patterns?
  - Did you choose numbers (messages) in ways to get new information?
  - Do you choose them in a way that made you right more often?

Exercise (continued)

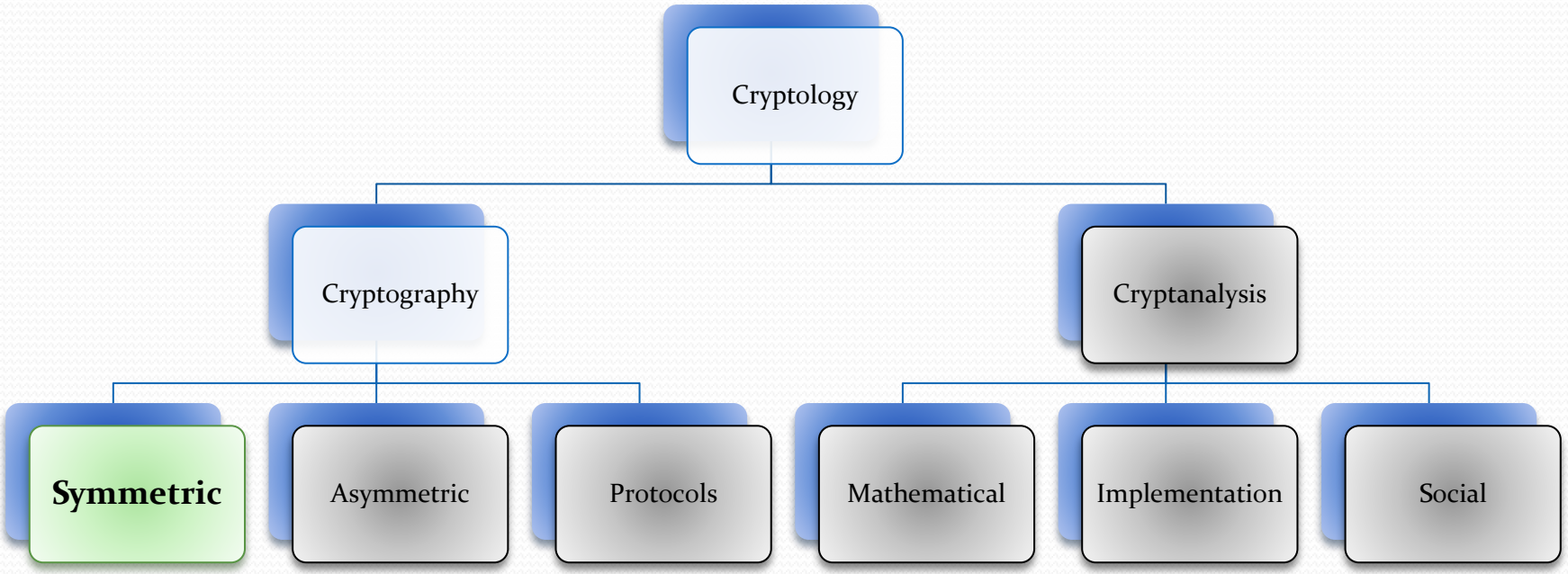
- If player 2 knows something useful (doesn't have to be the function, but a hunch) then they will have significantly more right or wrong after many rounds
  - Can distinguish ciphertext from noise
- If player 2 knows nothing, they will have about as many right as wrong after many rounds
  - No better than flipping a coin



Section Summary

- Cryptology is the study of codes
  - Cryptography is the study of designing codes
  - Cryptanalysis is the study of breaking codes
- Cryptanalysis plays a key role in algorithm design
- Encryption algorithms generally fall into two categories
  - Symmetric
  - Asymmetric

Symmetric cryptography



Symmetric algorithms

- The sending and receiving party use the same key
  - Alice computes y = Encrypt(x,k)
  - Alice sends y to Bob
  - Bob computes x' = Decrypt(y,k)

- The Caesar cipher was an example of symmetric cryptography

Types of symmetric crypto

- Symmetric encryption algorithms fall into two categories
  - Block ciphers
  - Stream ciphers

- Used for different purposes
- Both must provide confusion and diffusion

Confusion and Diffusion

- Claude Shannon was a famous information theorist
- Defined two properties that are widely used in symmetric cryptographic primitives

- Confusion
  - Relationship between key and ciphertext is obscured

- Diffusion
  - The influence of one plaintext symbol is spread over many ciphertext symbols with the goal of hiding statistical properties of the plaintext
    - Adversary must do more work to find statistical properties

Exercise

- Form groups of 4-6 people
- Each group should
  - Come up with a one-sentence message
  - Choose a key in A-Z (0-25)
  - Encrypt the message with the Caesar cipher
  - Exchange your ciphertext with another group
  - Find the message the other group wrote

Another Look at the Caesar Cipher

- How does the Caesar cipher stack up in terms of confusion and diffusion?

- Confusion
  - Yes

- Diffusion
  - Changing one symbol in the plaintext has a very predictable result
  - Only changes one symbol in the output

One-time pad

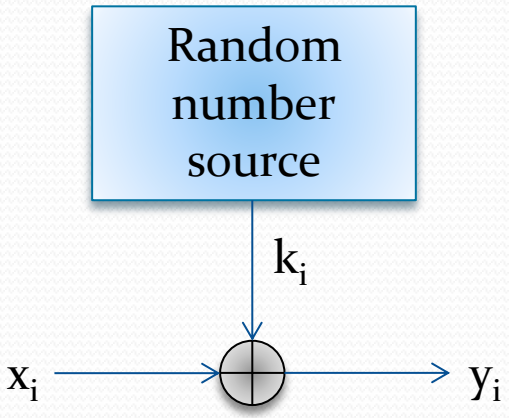
- One-time pad (OTP) is the only cryptosystem that achieves perfect secrecy
  - Given ciphertext *c*, plaintext message *m*
  - Random variables X and Y (for plaintext and ciphertext, respectively)
  - $\Pr[X=m|Y=c] = \Pr[X=m]$
    - The a posteriori probability that the plaintext is *m* is equal to the a priori probability that the plaintext is *m*
    - In other words, knowing the ciphertext doesn't give you any additional insight into the value of the plaintext

One-time pad (continued)

- OTP is also unconditionally secure
  - It cannot be broken even with *infinite* computational resources
- An attacker with infinite resources can break a 10,000-bit key cipher in one time-step
  - Have $2^{10,000}$ computers each try a key
  - That's more computers than atoms in the universe!
  - System is computationally secure
    - Adversary is computationally bounded

One-time pad (continued)

- How does this miraculous cryptosystem work?
  - Plaintext *m* = $x_0x_1\ldots x_{n-2}x_{n-1}$ in binary
  - Ciphertext c = $y_0y_1\ldots y_{n-2}y_{n-1}$ in binary
  - Key stream K = $k_0k_1\ldots k_{n-2}k_{n-1}$ in binary



Why doesn't everyone use OTP?

- The key bits need to be truly random
  - Does your computer have true random number generator? Mine doesn't
- The sender and receiver must have the same key stream
  - How do you communicate the key stream securely?
- A bit of the key stream can only be used once
  - Key needs to be as long as the message
  - That's a lot of bits over time
    - A lot to have to send securely

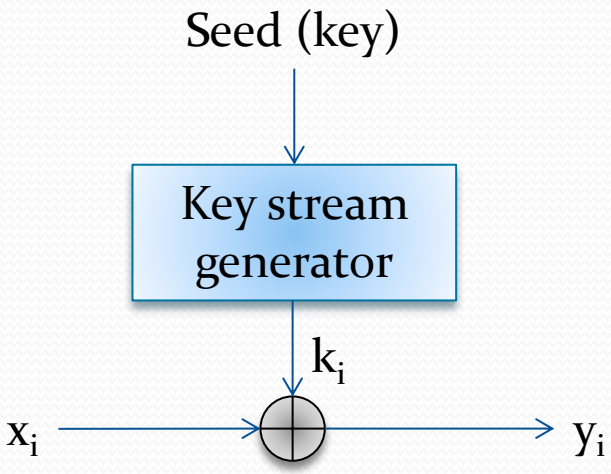
- Bottom line: doable, but not practical for the vast majority of applications

The good news

- Fortunately, we can approximate a OTP with a stream cipher

- Main idea:
  - Use a shorter key to generate a key stream in a pseudorandom fashion
    - Practical
    - Does not achieve perfect secrecy
    - Can achieve computational security

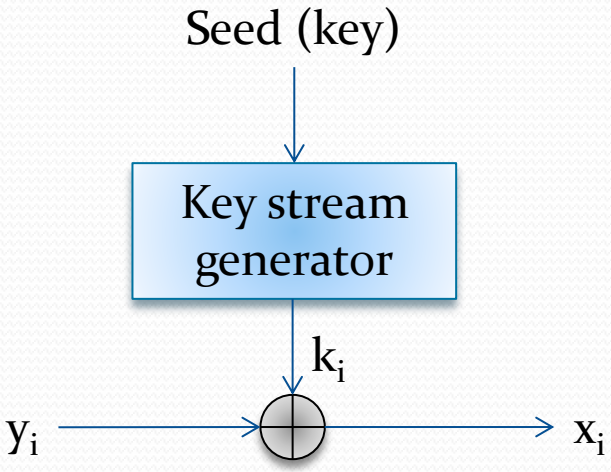
Stream cipher



- Message $m = x_0x_1\ldots x_{n-2}x_{n-1}$ in binary
- Each x_i is combined with a bit of the key stream via exclusive-or (addition modulo 2)

Stream cipher

- Decryption is the same as encryption
  - Swap plaintext and ciphertext



Example

- Suppose a stream cipher with the following recurrence:
  - $x_{i+4} = x_i + x_{i+1} \bmod 2$
  - $i \geq 1$

- Suppose that the initial values are (1,0,0,0)

Example (continued)

- 10001<u>1</u>
  - $x_5 = x_1 + x_2 \bmod 2$
- 10001<u>0</u>
  - $x_6 = x_2 + x_3 \bmod 2$
- 100010<u>0</u>
  - $x_7 = x_3 + x_4 \bmod 2$
- 10001001<u></u>
  - $x_8 = x_4 + x_5 \bmod 2$
- 100010011<u></u>
  - $x_9 = x_5 + x_6 \bmod 2$
- 1000100110<u></u>
  - $x_{10} = x_6 + x_7 \bmod 2$
- 10001001101<u></u>
  - $x_{11} = x_7 + x_8 \bmod 2$
- 100010011010<u></u>
  - $x_{12} = x_8 + x_9 \bmod 2$
- 1000100110101<u></u>
  - $x_{13} = x_9 + x_{10} \bmod 2$
- 10001001101011<u></u>
  - $x_{14} = x_{10} + x_{11} \bmod 2$

Example (continued)

- So we have a stream that looks like 1000100110101l…
- Suppose we want to encrypt 10101010
  - Combine message and key stream with addition modulo 2 (exclusive-or)

Message	1	0	1	0	1	0	1	0
Key stream	1	0	0	0	1	0	0	1
Ciphertext	0	0	1	0	0	0	1	1

Security

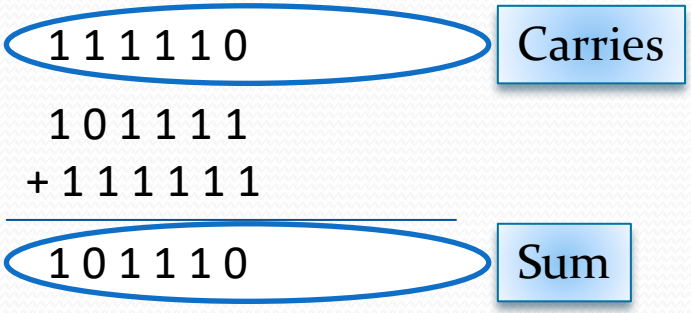
- That example was not secure, because the key stream was too predictable

Do you have to use xor?

- Well… no. But it has nice properties and is most common
  - Fast
    - No carries to deal with
  - Same function for encryption and decryption
    - Don't need a separate stream or operation to decrypt
- This is perhaps best shown by example

Addition mod m

- Instead of using addition mod 2, let's generalize to addition mod m, m > 2
- There will be lg(m)-1 carry values in the computation
- Concrete example: $m=2^7$



- Bitwise operations are convenient because we can skip a register

Other benefits of XOR

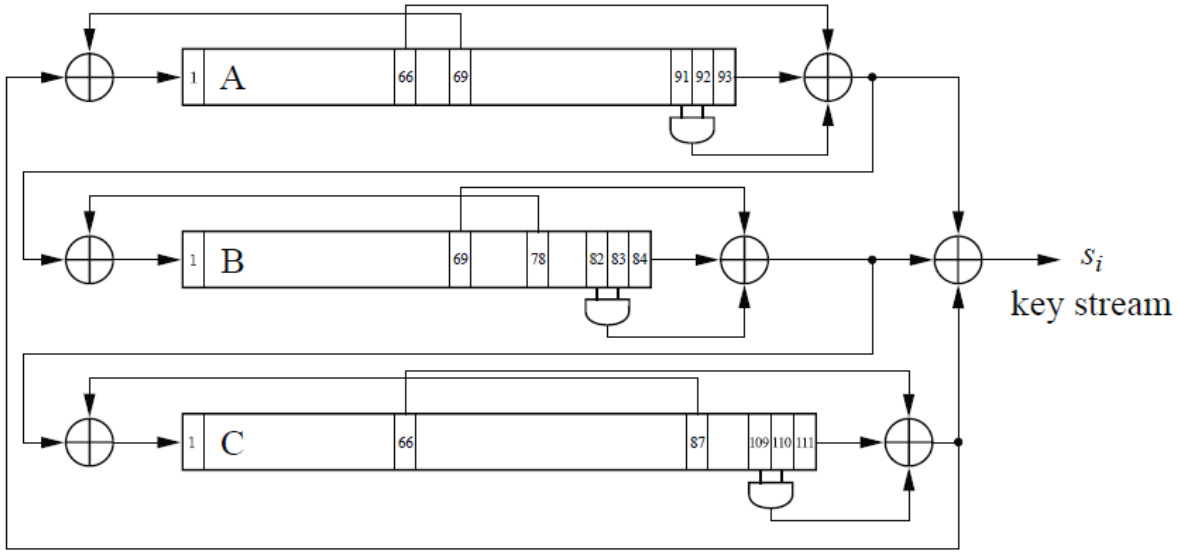
- Decryption performed by inverting combining operation
- The inverse operation of addition mod m is subtraction mod m
- The inverse operation of multiplication mod m is multiplication mod m, but you would need a different key stream
- The inverse operation of XOR is XOR
  - a XOR b = c
  - c XOR b = a

So how can a stream cipher be secure?

- Like the OTP, the security lies in the randomness properties of the key stream
  - Don't use the same bits of the key stream more than once, or else
    - a XOR k = a'
    - b XOR k = b'
    - a' XOR b' = a XOR k XOR b XOR k = a XOR b
      - Assume adversary knows a' and b'
      - Now a much smaller plaintext space for a and b
      - Can use heuristics to determine a and b
      - Gives adversary an advantage

Practical stream ciphers

- Large state size, large input key (>112 bits)
- Function that updates state is complicated and nonlinear
- State outputs few bits at each update (preferably one)
- In use
  - RC4
  - Snow
  - Salsa
  - Trivium ←



Question

- Does anyone happen to know what the US standard stream cipher is?

Answer

- That was a trick question
- We don't have one

- We might in the not-too-distant future

Why isn't there a US standard stream cipher?

- There hasn't been an outcry for one
- Block ciphers are more studied and better understood

- NIST is currently looking into this, but it is a new project

Block ciphers

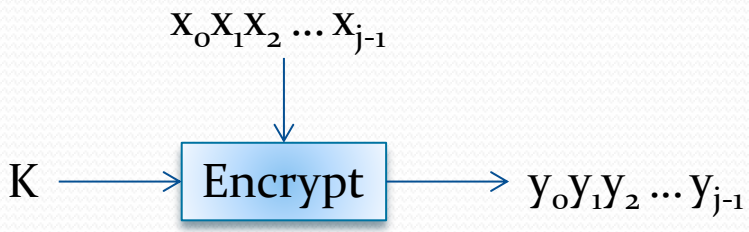
- Main idea: instead of encrypting one bit at a time, encrypt several bits together
- The grouping of bits is called a block

- Block ciphers play a significant role in many cryptographic systems

- Let's start by considering operations on a single block

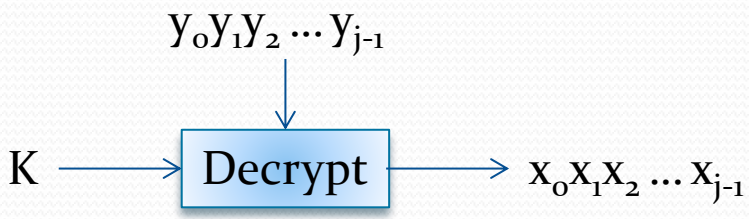
Block cipher encryption

- Plaintext $m = x_0x_1\ldots x_{n-2}x_{n-1}$ in binary
- Ciphertext $c = y_0y_1\ldots y_{n-2}y_{n-1}$ in binary
- Key K
- Block size = j



Block cipher decryption

- To decrypt, need to apply (possibly different) decryption algorithm



Block vs. stream

- Constrained devices
  - Stream cipher
- Real-time streaming data
  - If packets are multiple of block length, block cipher won't incur too high a penalty
  - If packet size varies widely, stream cipher may be better bet
- Everything else
  - Block cipher is probably best bet

An essential tool

- Block ciphers are not only used for encryption
- They are used to construct
  - Stream ciphers
  - Pseudorandom number generators
  - Hash functions
  - Message authentication codes

We'll get to these later

- Bottom line: block ciphers are an essential tool in symmetric cryptography

More than a block

- What happens when you need to encrypt more than one block?

Mode of operation

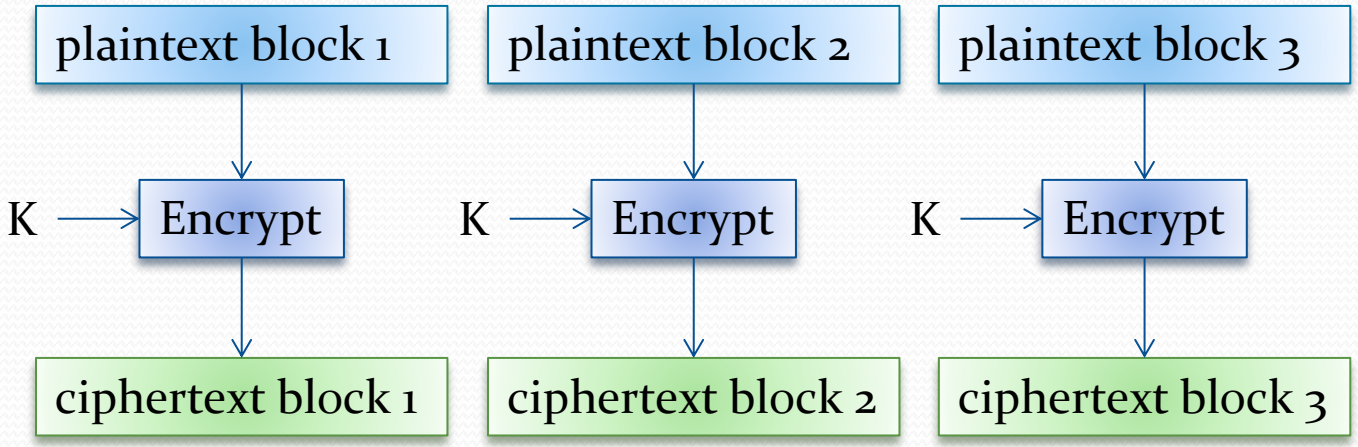
- Block ciphers are used in conjunction with a mode of operation
- Mode of operation determines how the blocks connect
- Right cipher + wrong mode = wrong solution

- We'll discuss
  - Electronic Codebook mode (ECB)
  - Cipher Block Chaining mode (CBC)
  - Cipher Feedback mode (CFB)
  - Output Feedback mode (OFB)
  - Counter mode (CTR)

ECB

- The most basic mode
- Each block is encrypted independently

- Let's look at the pros and cons

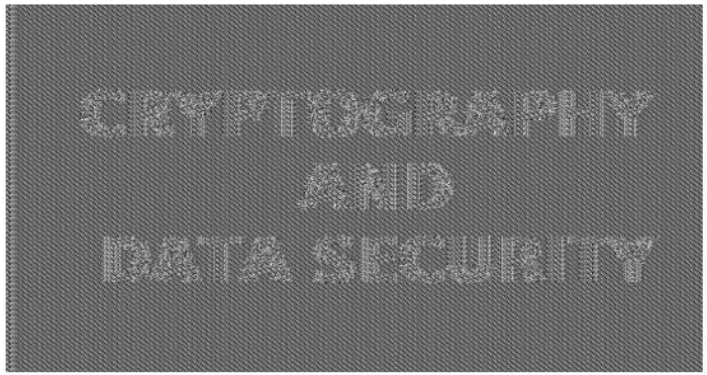


ECB: pros

- An error in one block will not affect others
  - Transmission errors
  - Missing blocks
- Easily parallelizable
  - Efficient to compute

ECB: cons

- Leaks information about repeating blocks
  - Repeating plaintext results in repeated ciphertext
- Adversary can rearrange message
  - “attack don’t stop!” vs. “stop! don’t attack”
- Adversary can substitute blocks



ECB example: block substitution

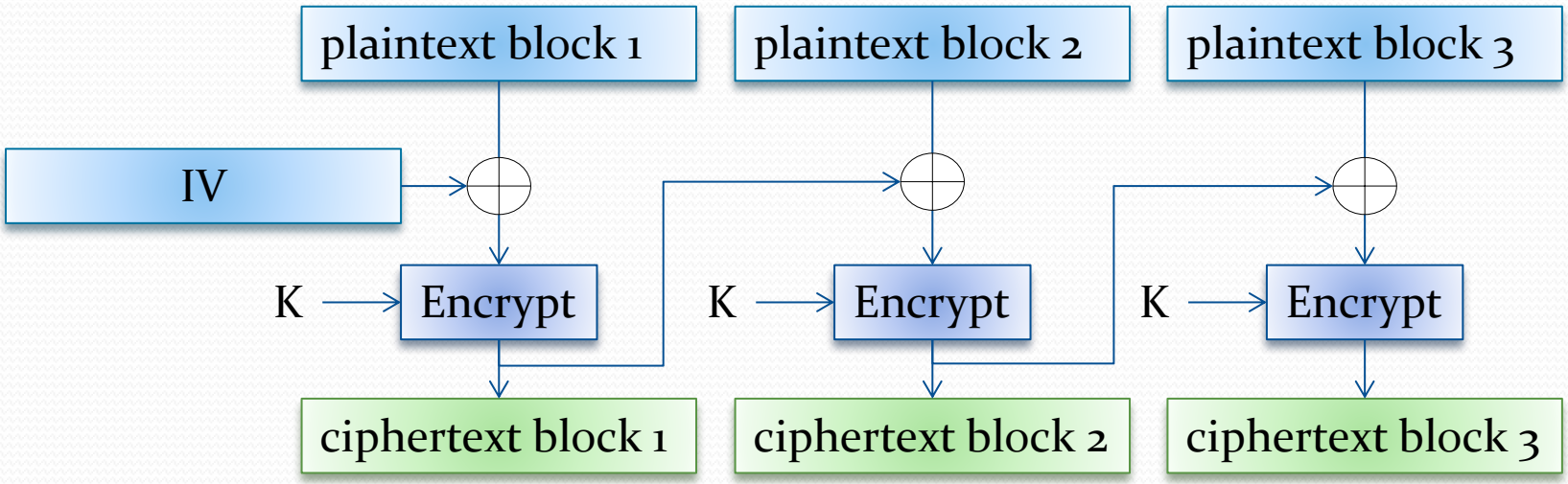
- Suppose the following blocks are part of a banking protocol

Sending bank	Sending account #	Receiving bank	Receiving account #	Amount $

- Sending and receiving banks share a key
- Adversary
  - Opens accounts at each bank
  - Sends money from one to the other, captures ciphertext
    - Now can identify transfers between the two banks
  - When other transfers go between those banks, replaces receiving account # with his
  - Gets lots of money
  - Withdraws money and commences with getaway plan

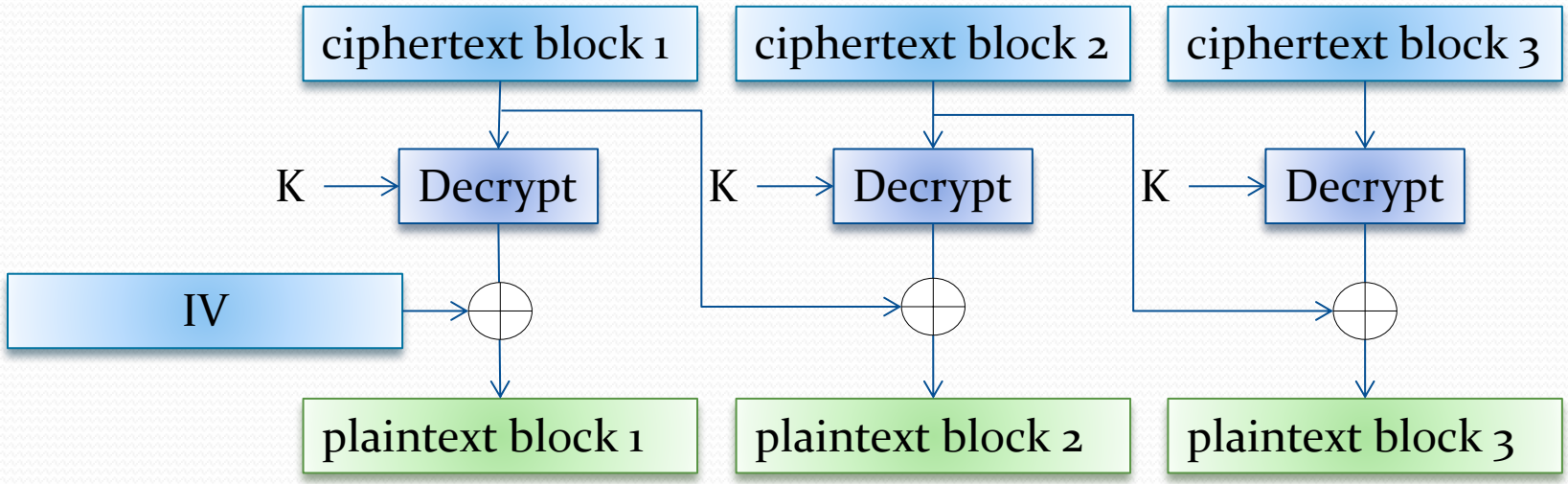
CBC

- Most common
- Blocks dependent on previous blocks
- Requires an initialization vector (IV)



CBC: decryption

- Encryption: XOR then encrypt
- Decryption: decrypt then XOR



CBC: a few more words on IV

- IV is same size as block
- IV is not a secret
  - Sent in the clear with ciphertext
    - Necessary for decryption
- Think twice about using predictable IV values
  - Examples
    - Counters
    - Repeated values
    - Using last ciphertext block of previous encrypted packet
  - Advanced attacks exploit predictability
    - E.g. BEAST
- Use random IV for each encryption
  - This is important
  - Random IV is necessary for security properties to hold

CBC: error tolerance

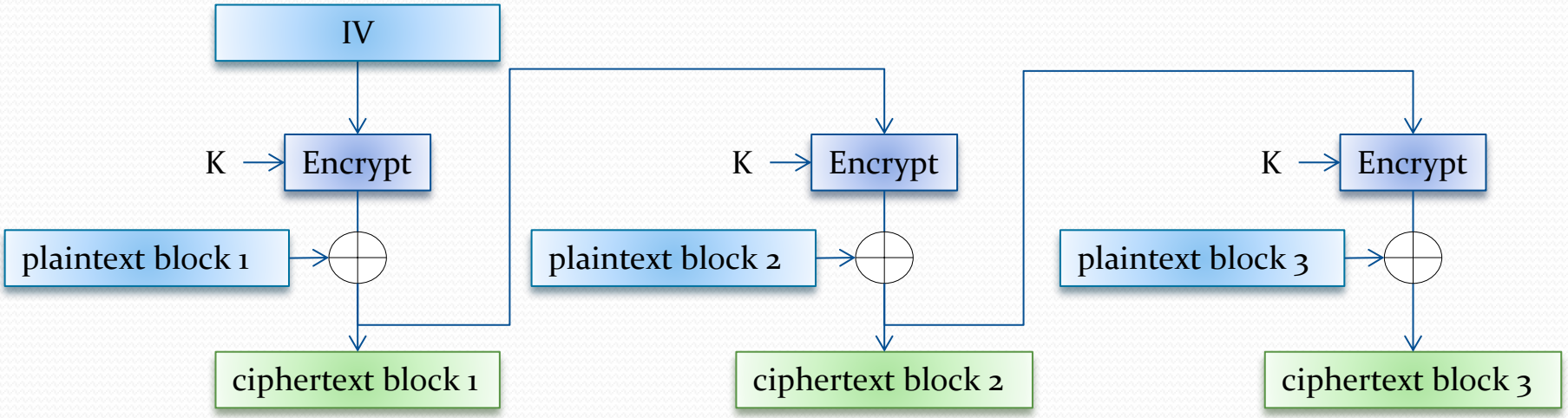
- Encryption
  - Encryption errors propagate forward
    - Each block uses the result of the previous block
    - If there was an error in encrypting block 1, that error affects block 2, block 3, and so on
- Decryption
  - Wrong IV in decryption only corrupts first block
    - Again, don't consider IV secret
  - Change to a block of ciphertext corrupts decryption of only two blocks
  - Adding blocks to the end of ciphertext won't alter anything before
    - Probably decrypt to rubbish, but there's a chance it won't

CBC: pros & cons

- Pros
  - IV makes encryption probabilistic
    - Same message-key pair map to different values when IVs are different
  - More difficult to attack through substitution
  - Decryption can be parallelized
- Cons
  - Encryption sequential
    - Cannot be parallelized
  - Encryption errors propagate
    - Bit flip early on messes up a lot
- Ciphertext stealing may be pro or con
  - Does not require message expansion (padding)

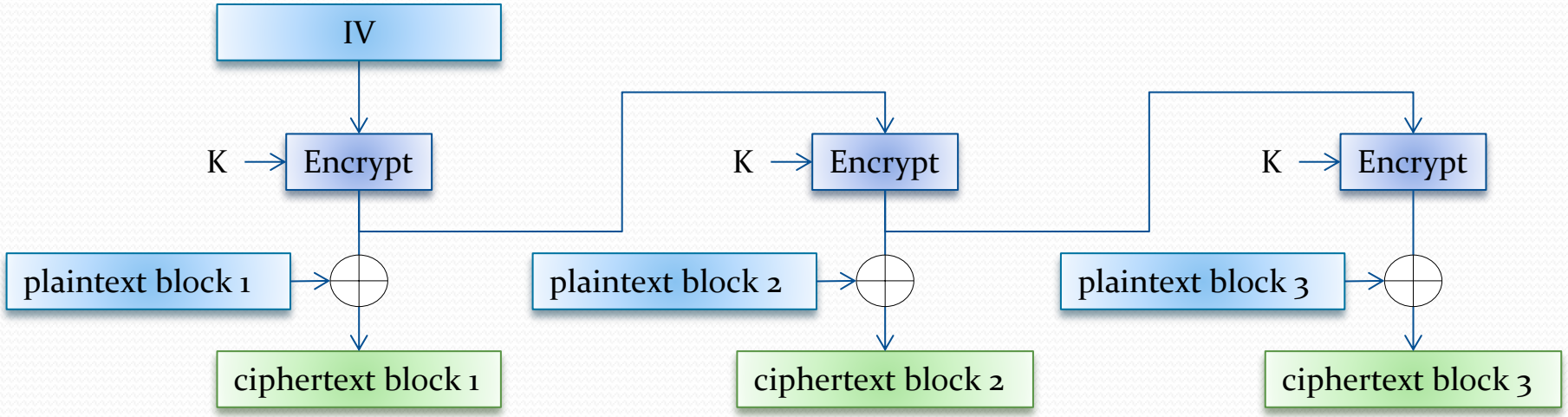
CFB

- Stream cipher–like functionality from a block cipher
- IV has to be unpredictable
- Decryption is same operation as encryption



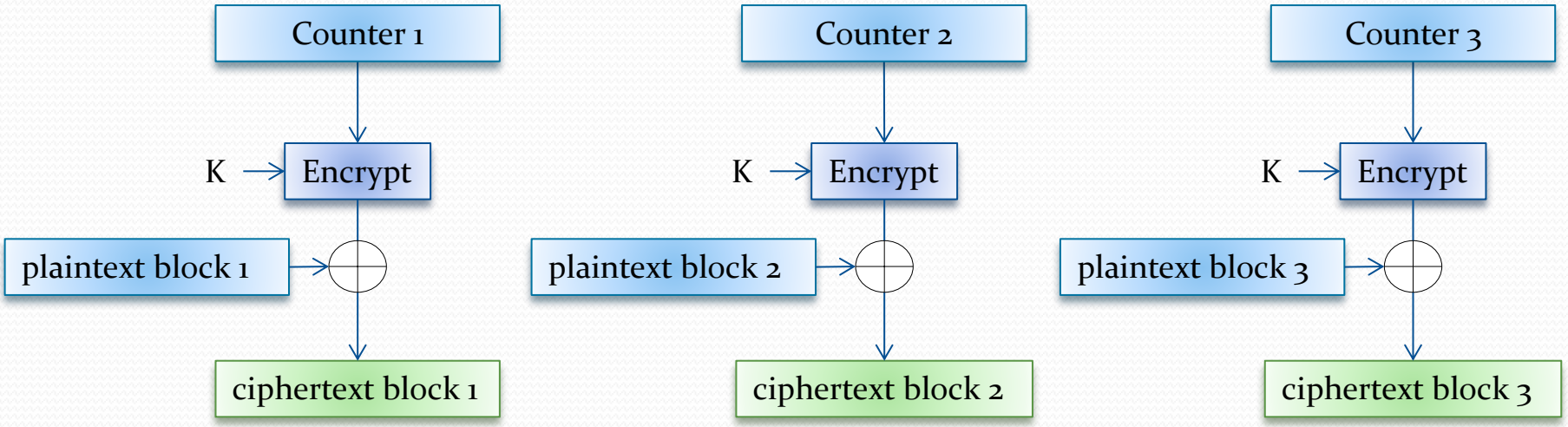
OFB

- Similar to CFB
- Turns block cipher into a stream cipher
- IV has to be unique, but not necessarily random



CTR

- Different variants
  - Talk about one in NIST SP 800-38a
- Turns block cipher into stream cipher
- Counter increases with each block
  - Does not need to be secret
  - Needs to be unique across blocks



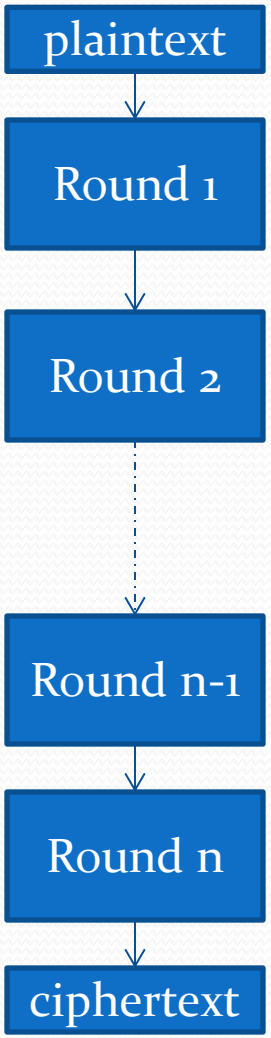
Choosing a mode

- General knowledge
  - CBC = good, ECB = bad
- It isn't this simple
  - Everything depends on context
  - There are more modes to consider
    - Others we talked about and more that we didn't
  - What are your requirements?
    - Parallelizable? Error tolerant? Synchronizing? Memory, speed, energy requirements? Malleability?
- It's difficult to generalize
  - What is right for one situation may be wrong for another

Block cipher construction

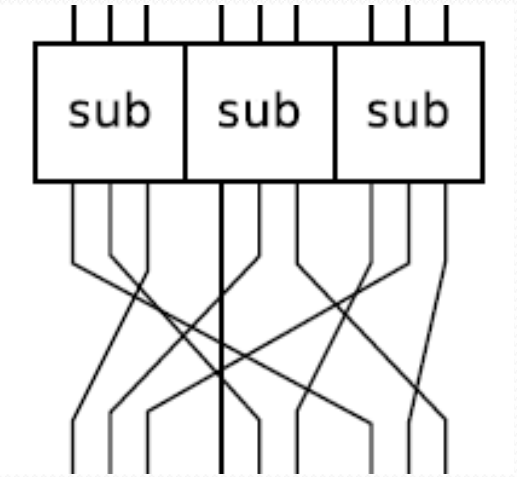
Common structure

- Product cipher
  - Combines two or more transformations
  - Resulting cipher more secure than components
- Round function
  - Provides confusion and diffusion
  - Key combined with state (confusion)
  - Data mixing/permutations (diffusion)



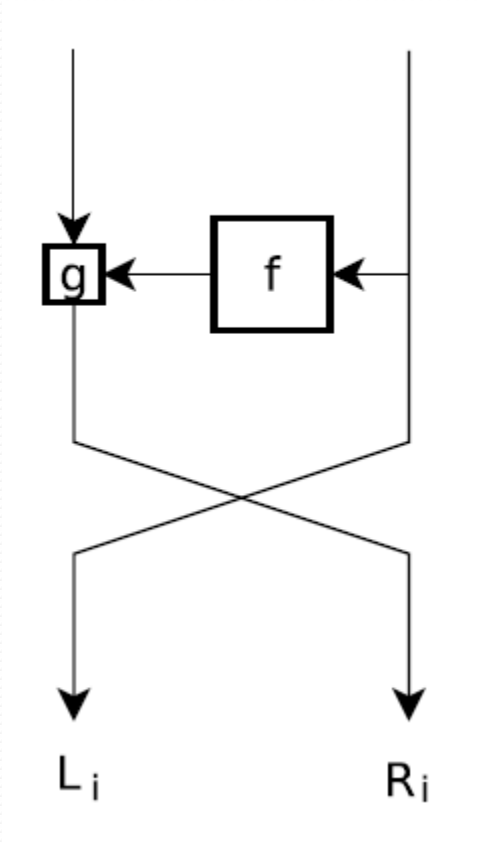
Round structure: SPN

- Substitution Permutation Network (SPN)
- Substitution box (S-box)
  - Invertible non-linear function
  - Often implemented as look-up tables
- Permutation
  - Change bit locations



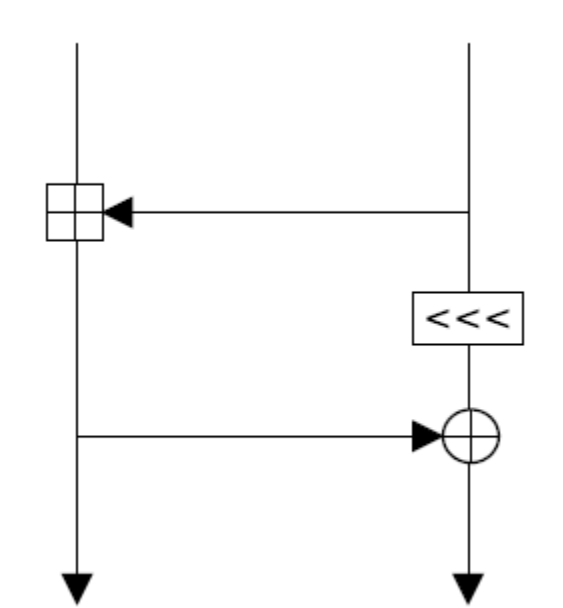
Round structure: Feistel

- Split state into left and right
- Encrypt and decrypt functions use same logic
  - Smaller footprint
- f contains a non-linear function
  - S-boxes do not need to be invertible in Feistel ciphers
- g is operation to combine the left and right
  - Example: xor



ARX

- Addition-Rotation-XOR
- Two kinds of linear operations
  - Translating them into a single linear form is difficult over many rounds
- Acts as simple substitution box
  - No need for tables
  - Easy to compute



More rounds = more security, right?

- Not exactly
- While this seems intuitive, there is a caveat
  - Rounds cannot be exactly identical
  - Tools for this
    - Key schedule
    - Round constants

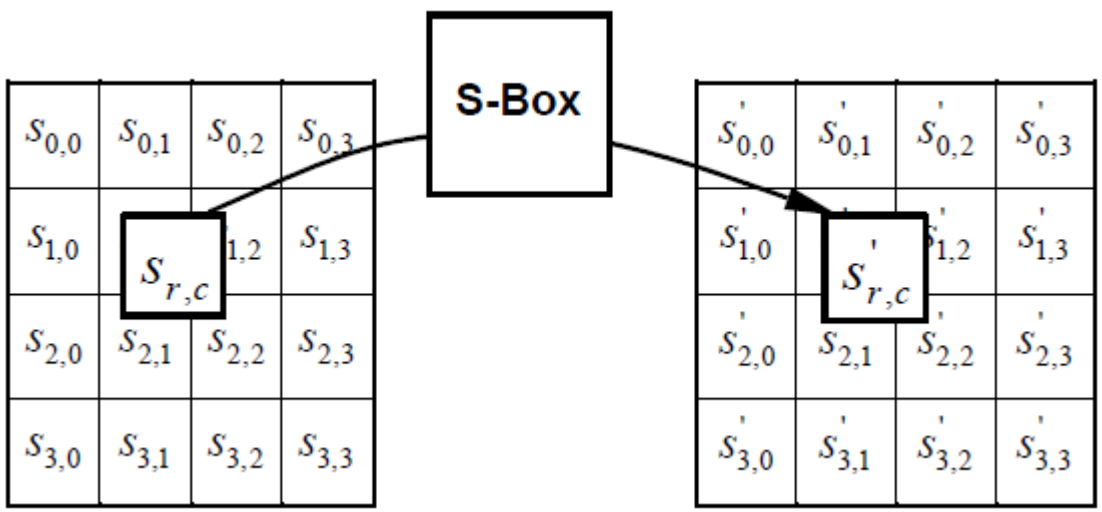
Concrete example: AES

- Let's look at the round components of a real cipher
- Round consists of nonlinear step and linear mixing
  - Which provides confusion, and which provide diffusion?

- Round structure
  - SubBytes
    - nonlinearity
  - ShiftRows, MixColumn
    - Linear mixing
  - AddKey
    - Combine key with state via exclusive-or

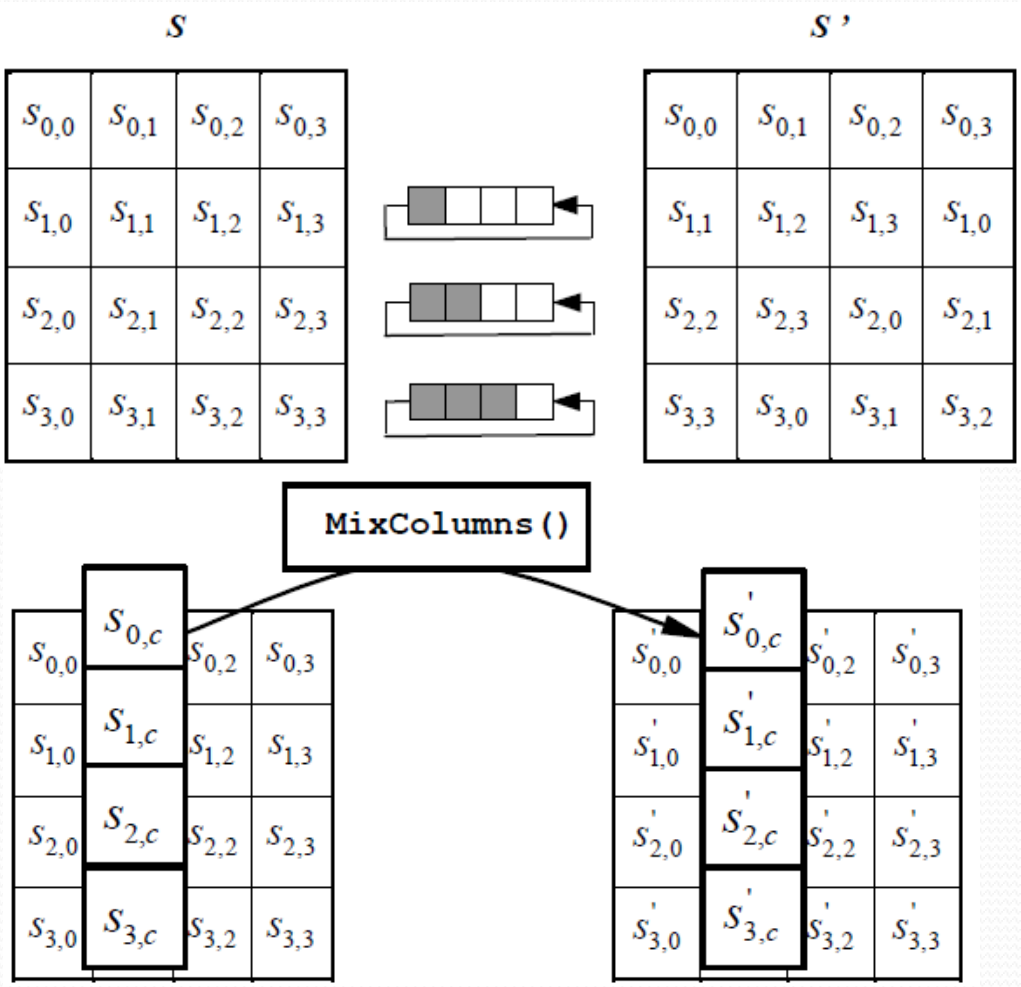
AES: SubBytes

- Byte substitution
  - Replace a byte of data with another according to a function
  - The function is invertible so that we can reverse the substitution
  - $S'_{r,c} = \text{S-Box}(S_{r,c})$

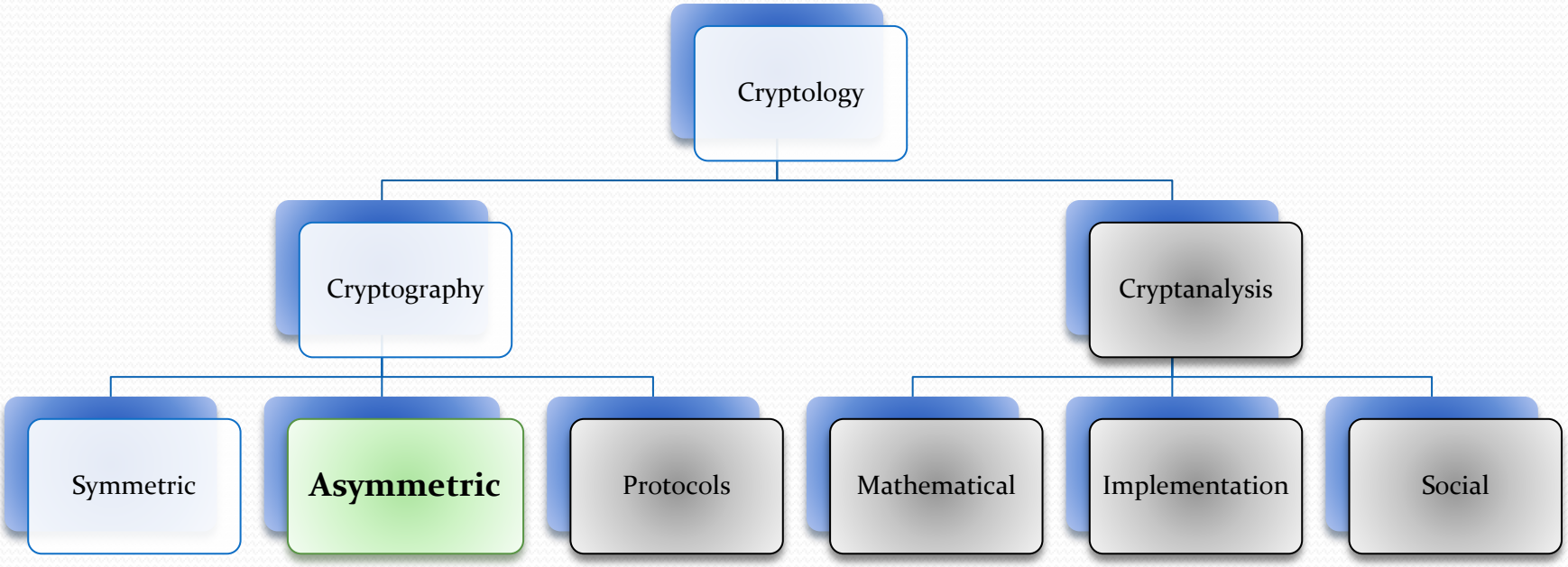


AES: ShiftRows and MixColumns

- ShiftRows
  - Circular shift on each row of the state
    - Shift amount differs by row
- MixColumns
  - Multiply each column of the state by a fixed matrix
  - $S'_{*,c} = S_{*,c} \cdot \text{matrix}$



Asymmetric Cryptography

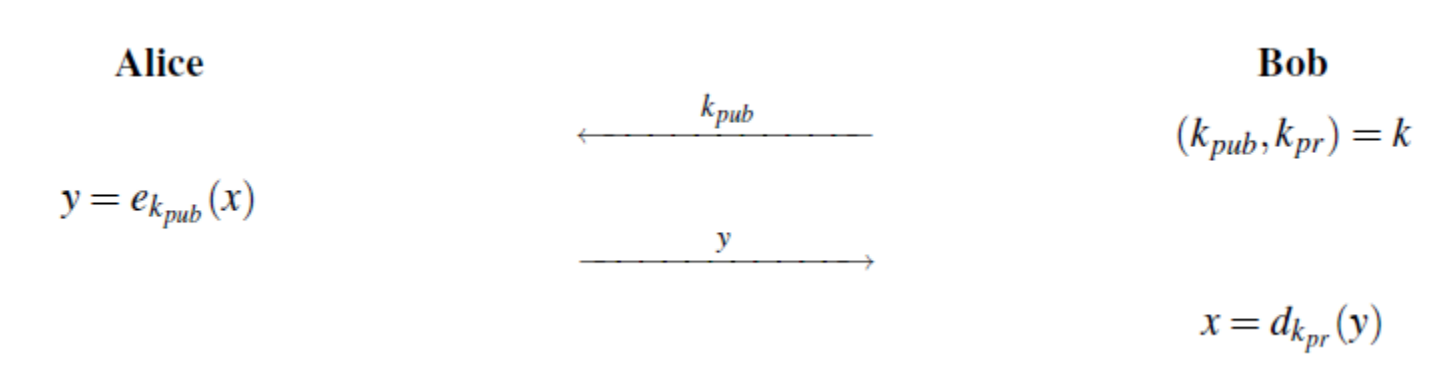


Asymmetric Crypto

- Asymmetric cryptography = public key cryptography
- Different key used by different parties
  - Key use is not symmetric

- Mailbox analogy
  - There is a locked mailbox with a slot
  - Anyone can put a message in the box
  - Only a person with the mailbox key can retrieve the messages

The Basic Picture



- Bob gives Alice his public key, k_{pub}
- Alice can send Bob *x*, encrypted with Bob's public key
- Only a person who knows Bob's private key can decrypt it and get *x*

Principles of asymmetric ciphers

- Based on hard problems
  - Not confusion and diffusion
- As long as there is no polynomial-time algorithm to solve underlying problem, algorithm secure
  - *with reasonable key length

Factoring

- Given an integer n, find the prime factorization
  - 42 = 2 * 21 = 2 * 3 * 7
- Now do it for a 2048-bit integer
  - Yea… it is tough
- Trapdoor function that requires knowledge of factorization
  - Primes are part of the private key
  - Adversary only gets the product
    - Has a lot of work ahead

Factoring: RSA

- N=p*q, p and q are primes
- Encryption: x^e mod N
- Decryption: x^d mod N

- RSA public key has (e, N)
- Decryption exponent d derived from e, p and q

Discrete logarithm

- Given $x^a = y$ and x, what is a?
- There are conditions
  - Only certain set/operation combinations can be used for crypto
  - x must be a generator (x has to create all elements of set when you keep multiplying it by itself)

Post-quantum

- Solve factoring and you also solve discrete log
- Shor's algorithm does just that
  - The catch is that you need a quantum computer to run it
- Research in post-quantum asymmetric algorithms
  - Multivariate public key
    - The coefficients of polynomials are polynomials
  - Lattice-based cryptography
  - Code-based cryptography
    - Error-correcting codes

Symmetric Vs. Asymmetric

Symmetric	Asymmetric
Same key for encryption and decryption	Different keys for encryption and decryption
Based on confusion and diffusion	Based on number theoretic problems
Repeated iterations of simple function	Can be expressed as simple equation

When should I use what?

- Symmetric algorithms
  - More computationally efficient
  - Key distribution is a problem
- Asymmetric algorithms
  - More computationally expensive
  - Do not need to agree upon a key
  - Need to establish authenticity of public key
- Symmetric better for sending data
- Asymmetric better for sending symmetric keys
  - Wrap symmetric key

Key length

- Key lengths for asymmetric algorithms are different from symmetric
- In general, it takes a longer key to have the same protection against brute force attacks

Algorithm Family	Cryptosystems	Security Level (bit)			
		80	128	192	256
Integer factorization	RSA	1024 bit	3072 bit	7680 bit	15360 bit
Discrete logarithm	DH, DSA, Elgamal	1024 bit	3072 bit	7680 bit	15360 bit
Elliptic curves	ECDH, ECDSA	160 bit	256 bit	384 bit	512 bit
Symmetric-key	AES, 3DES	80 bit	128 bit	192 bit	256 bit

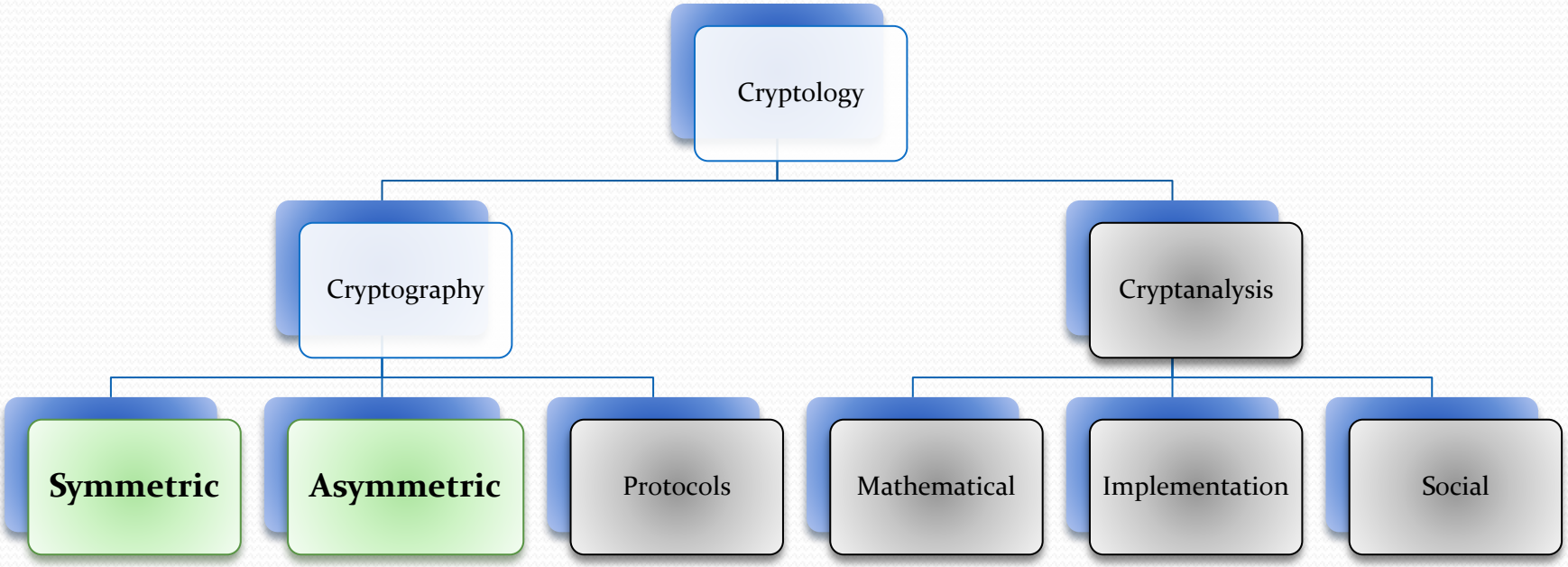
Note: This is no longer sufficient

Key length (continued)

- Why do you think that asymmetric algorithms require longer keys?

More primitives

Now that we've looked at symmetric and asymmetric topics, we'll dive a little further into both areas

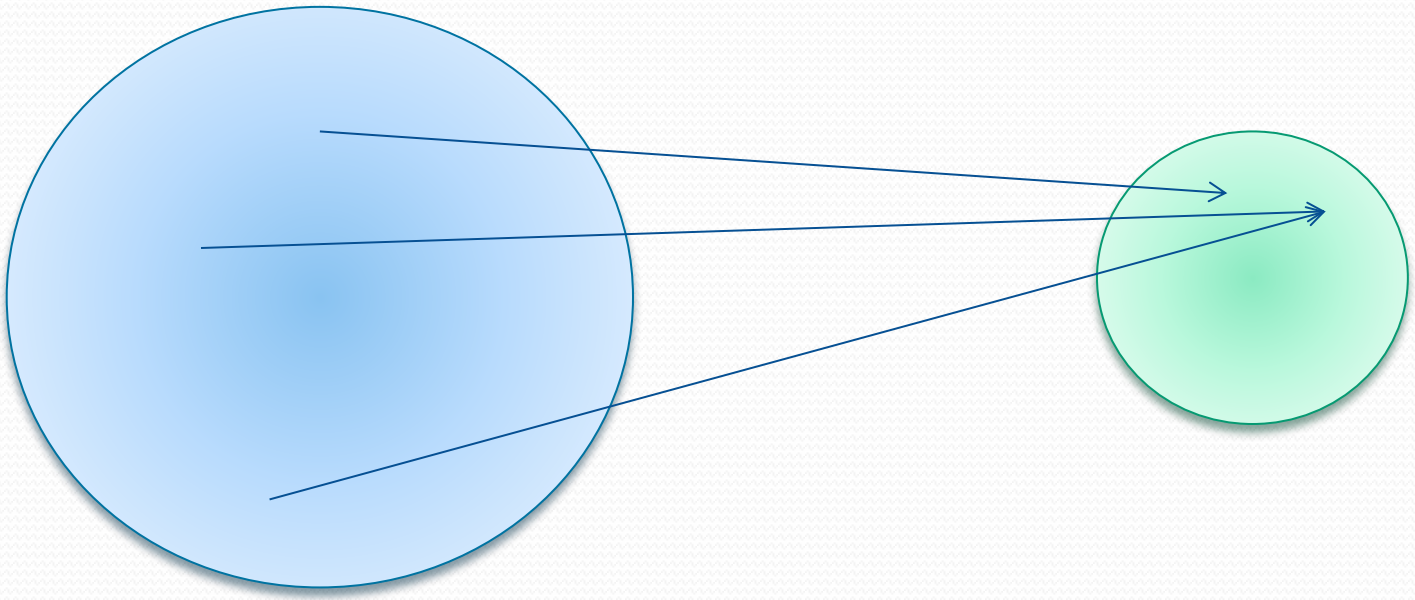


Hash functions

- Do not encrypt/decrypt
  - Do not provide confidentiality
- Do fall under symmetric cryptography
  - Constructed with symmetric primitives
- Provide means of integrity checking

What is a hash function?

- Function that maps arbitrary-length input to fixed size output
- Output space is smaller
  - Multiple inputs map to same output



Cryptographic hash functions

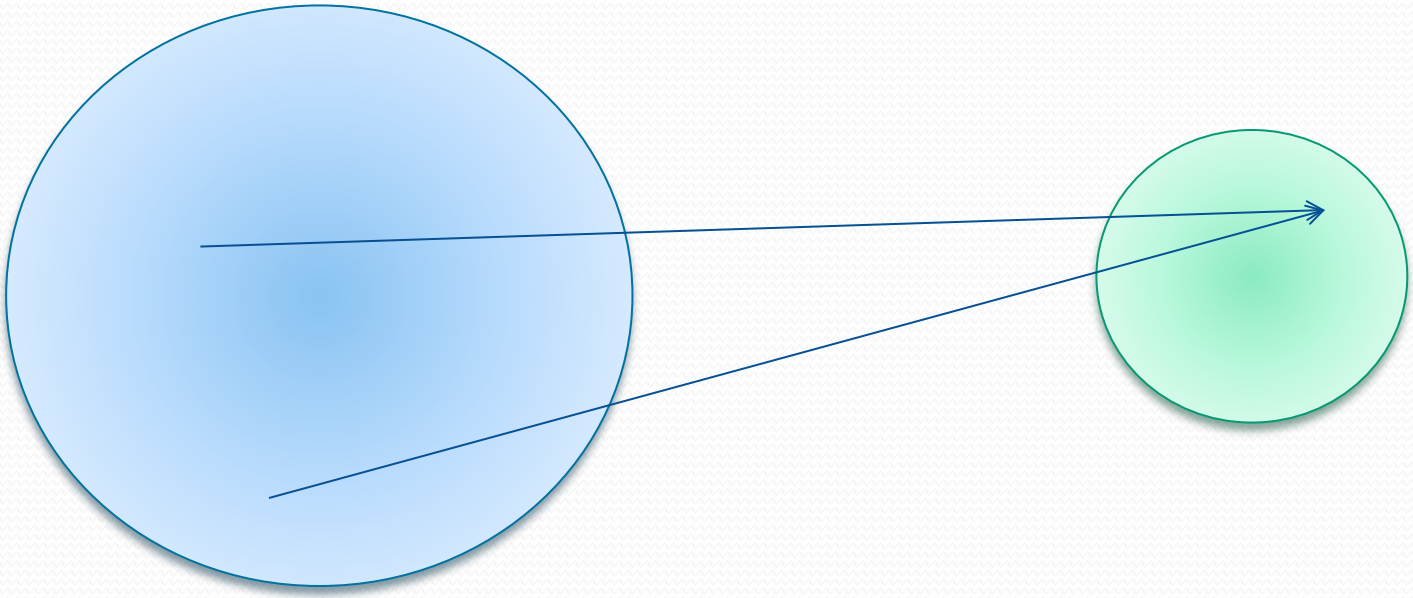
- When we say “hash functions”, we often mean “cryptographic hash functions”
- Create a “digital fingerprint” of data

- Three properties
  - Collision resistance (strong collision resistance)
  - Preimage resistance
  - Second preimage resistance (weak collision resistance)

- There are variants for each of these, but only main concept covered here

Collision resistance

- Infeasible to find $x \neq y$ such that $h(x) = h(y)$



- Resistance is upper-bounded by the birthday problem

Birthday problem

- Deals with probability that two people have the same birthday
- We have *n* people in this room
- There are 365 days in a year
  - 366 if February 29 included

- Probability of two of the same birthday is about 0.5 when n=23

Birthday problem (continued)

- Assume only 365 days
- Start with two people
  - Pr(2 people no same birthday) = 1 – 1/365
- Add a third person
  - Pr(3 people no same birthday) = (1 – 1/365)(1 - 2/365)
- Add *t* more people
  - Pr(*n* people no same birthday) =

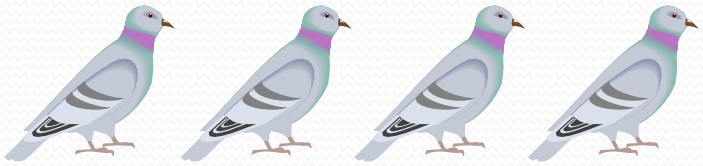
    (1 – 1/365)(1 - 2/365)…(1 – (*n*-1)/365)

Birthday problem (continued)

- Pr(at least one collision) = 1 – Pr(no collision)
  - $1 - (1 - 1/365)(1 - 2/365)\ldots(1 - (n-1)/365)$
- Let n=23
  - $1 - (1 - 1/365)(1 - 2/365)\ldots(1 - (23-1)/365) = 0.507 \approx 0.5$
- Let's try it with this group
- When can we guarantee a collision?
  - Pr(collision) = 1

Pigeon hole principle

- If n pigeons are put in m pigeon holes, $n > m$, then at least one hole contains more than one pigeon
  - Note that some holes may be empty



Pigeon hole and collisions

- If the output space size is m, then $m+1$ inputs needed to guarantee a collision

- Assuming 365 days, need 366 people to guarantee at least two share a birthday
  - Need 367 people if assuming 366 days

- For hash function with n-bit output ($m=2^n$), then 2^n+1 message blocks needed for collision

Birthday bound

- Output should be too large for *m+1* blocks to be feasible
- Birthday attack (based on birthday problem) bigger concern
  - Generic attack that works on any hash function
  - Not guaranteed to be best attack on any specific hash function
  - Provides the birthday bound

	Hash output length				
λ	128 bit	160 bit	256 bit	384 bit	512 bit
0.5	2^{65}	2^{81}	2^{129}	2^{193}	2^{257}
0.9	2^{67}	2^{82}	2^{130}	2^{194}	2^{258}

Recall that 80-bits is ok for now, but moving towards 112

Random oracle model

- Captures the "ideal" hash function
  - Give the oracle x
  - Oracle responds with a seemingly random value
  - Always responds in the same way for the same input
- Think of as looking up h(x) in a massive book of truly random numbers

- There is no such thing as a true random oracle
  - But we want our hash functions to act like one

Random oracle model (continued)

- We were thinking in this model just now
- "All birthdays not equally likely"
  - Probably true, but when we think in the ROM, they are
- $\Pr[h(x) = y] = 1/M$
  - Where M is the size of the output
- Knowing h(x) should give no insight about the result of any other value
  - Must query oracle to find h(x')

Random oracle model (continued)

- If a hash function can be reliably distinguished from a random oracle, then there is trouble
- In the ROM, interesting things can be proven
- Collision resistance implies preimage resistance

Exercise: random oracle

- Break into pairs
  - Decide who will be player 1 and who will be player 2
- Player 1 acts as oracle
  - Come up with a simple hash function (not cryptographic) on integers
    - Example: $h(x) = 2x+3 \bmod 5$
    - Make sure you have a modular operation to keep the output size fixed
  - Don't tell player 2 your function
- Player 2
  - Query the oracle with integers
  - Given the result of several queries, try to guess the results for other values you haven't queried with yet
    - Tell player 1, and they will tell you if you are right or wrong
  - Keep track of right and wrong
- Switch roles

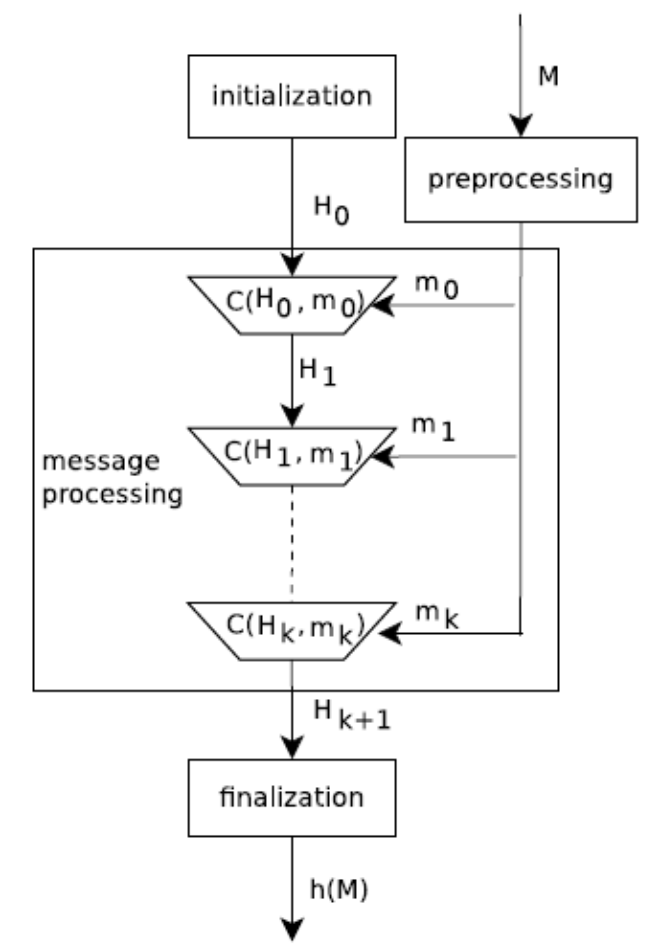
Exercise: random oracle (continued)

- It is helpful to make a table to capture what you've learned so far
- Based on this table, what do you think h(20) is?

x	h(x)
0	0
1	1
50	0
10	0
15	1

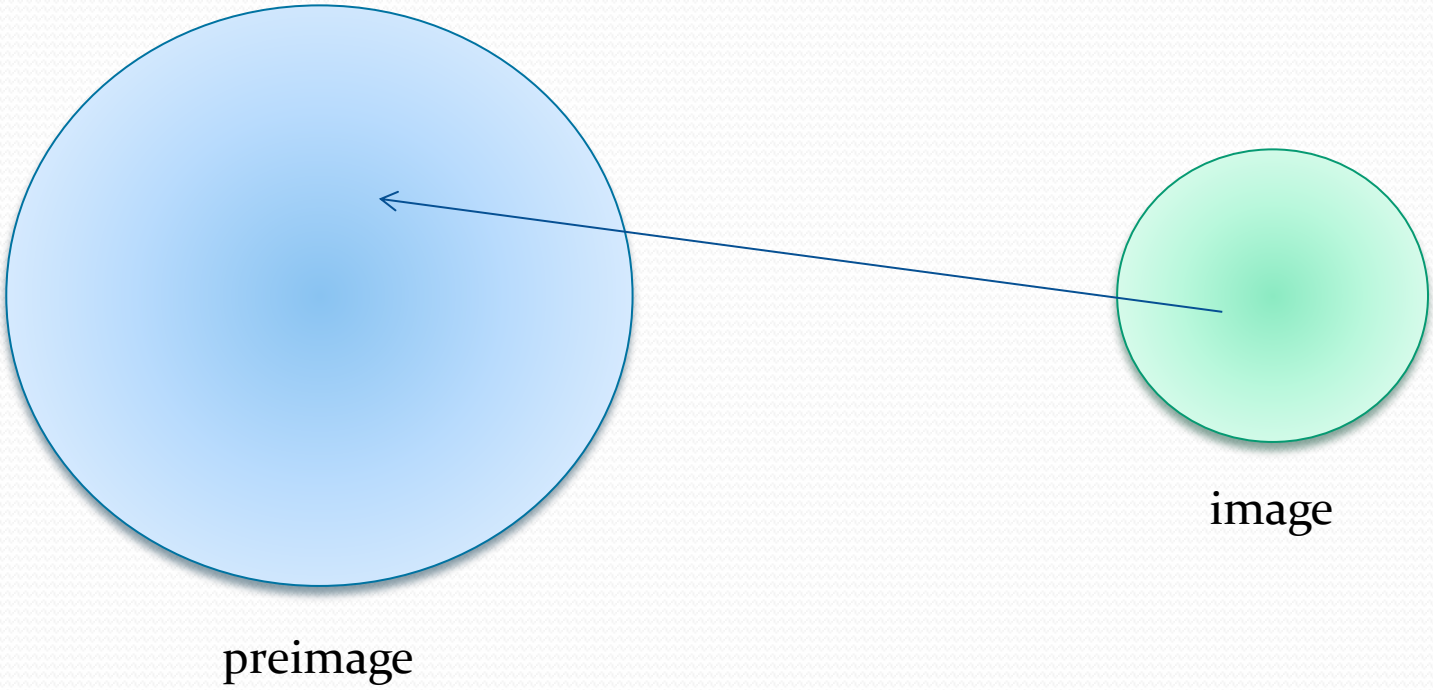
Iterated hash construction

- Fixed-size blocks as input
- Compression function acts as a random mapping
- Ingests a block of the message



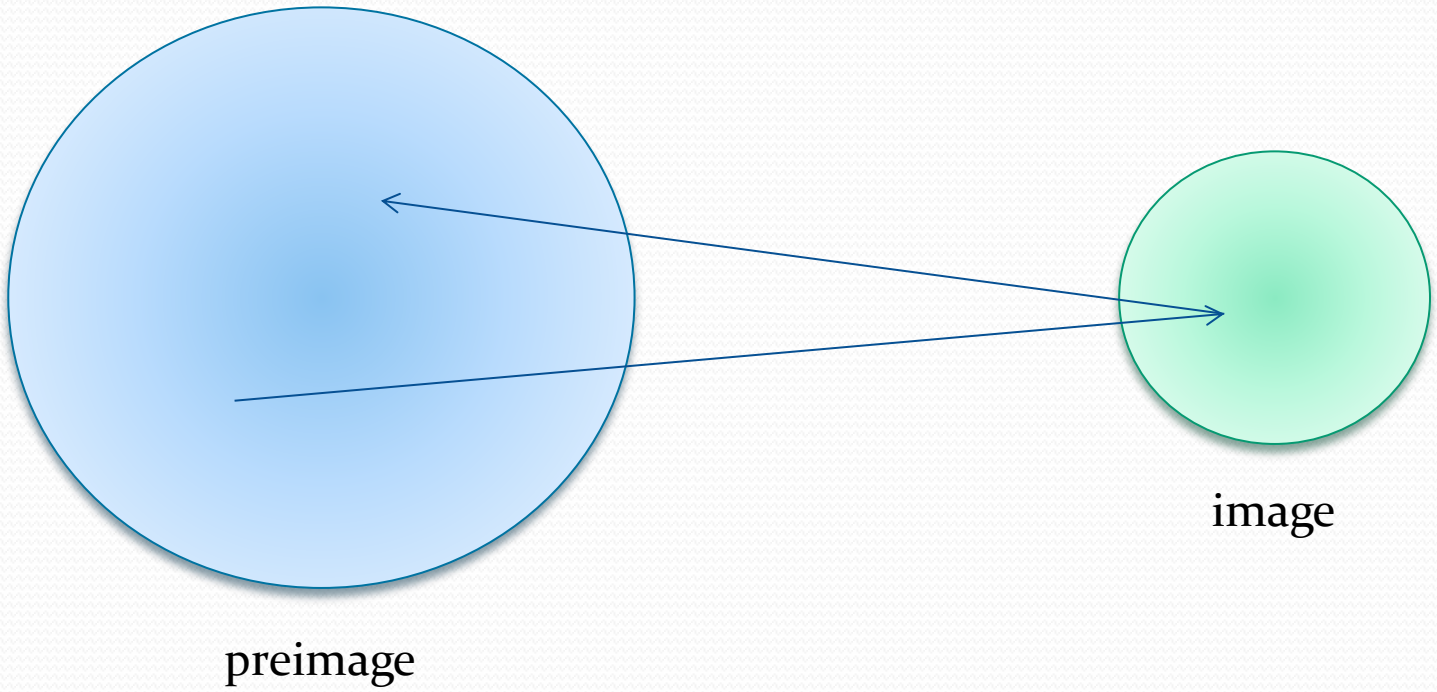
Preimage resistance

- Given z, infeasible to find x such that h(x) = z
  - Infeasible to find a value that maps to h(x)



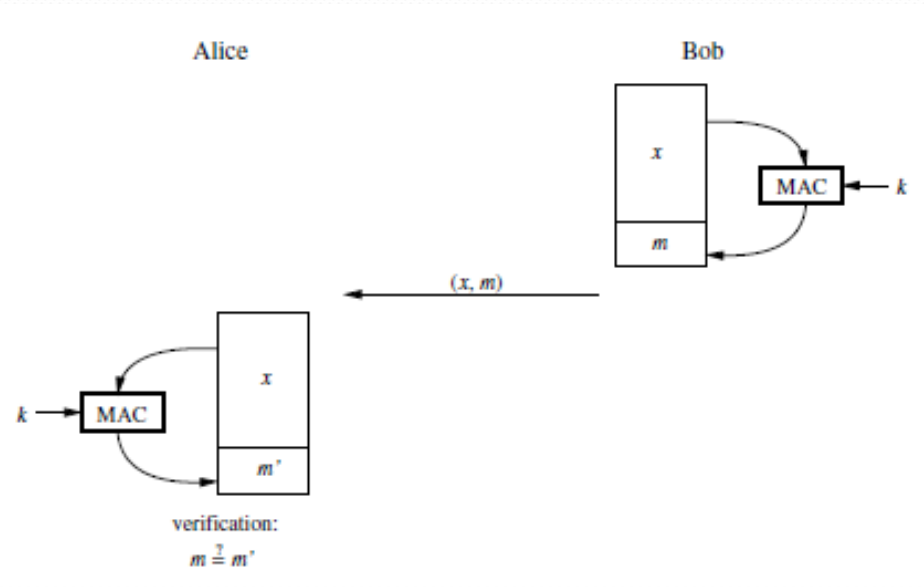
Second preimage resistance

- Given x, infeasible to find $y \neq x$ such that $h(x) = h(y)$
  - Infeasible to find second value that maps to $h(x)$



Message authentication codes

- Authentication
  - Someone who knows the key created this MAC
- Integrity
  - If the message is altered, the MAC will not match



MAC integrity

- Integrity + authentication
  - Can detect accidental or nefarious modifications

- Can have variable length MAC
- Fixed length MAC nicer
  - Sign a hash
    - Hash function provides additional integrity check

Who created the message?

- Alice, Bob, and Charlie share a key
- Message: “Charlie is stupid”
- Charlie is upset and wants to know who wrote it
- Alice says Bob did
- Bob says Alice did
- Because they both know the key, it could have been either of them
  - No definitive proof which one created it
- It may have even been Charlie, as far as Alice and Bob are concerned

Digital signatures

- In the real world, contracts are finalized with signatures
- Signing indicates that you agree to terms
- Signature assumed to be unique to individual
  - Legal and social barriers to prevent forgeries

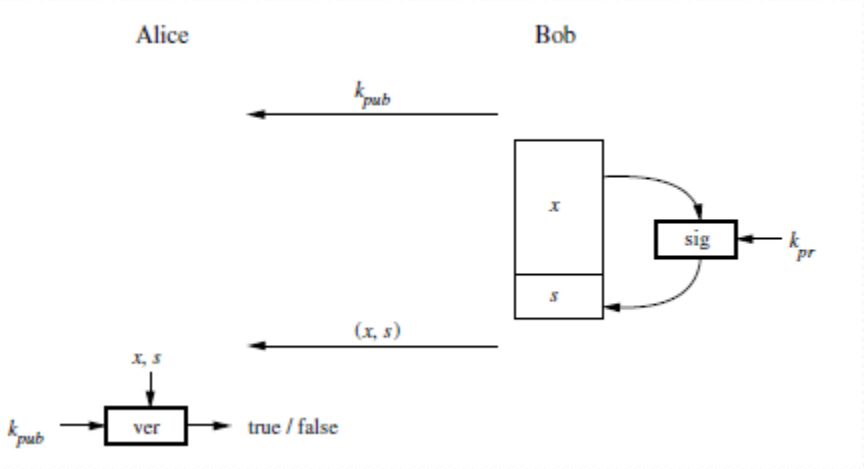
- Only the person who knows a key can generate a valid signature
  - Only one person knows that key (supposedly)
  - Authenticates signer

Digital signatures (continued)

- Uses asymmetric encryption and hash functions
- Sign with private key
  - Anyone can verify signature using public key
- But asymmetric operations expensive
  - What if message is longer than allowed?

How RSA digital signatures work

- Bob hashes plaintext message
  - Make fixed size input
- Sign (encrypt) hash with his private key
- Alice decrypts signature with Bob's public key
- Hashes message
- Verifies that the two match



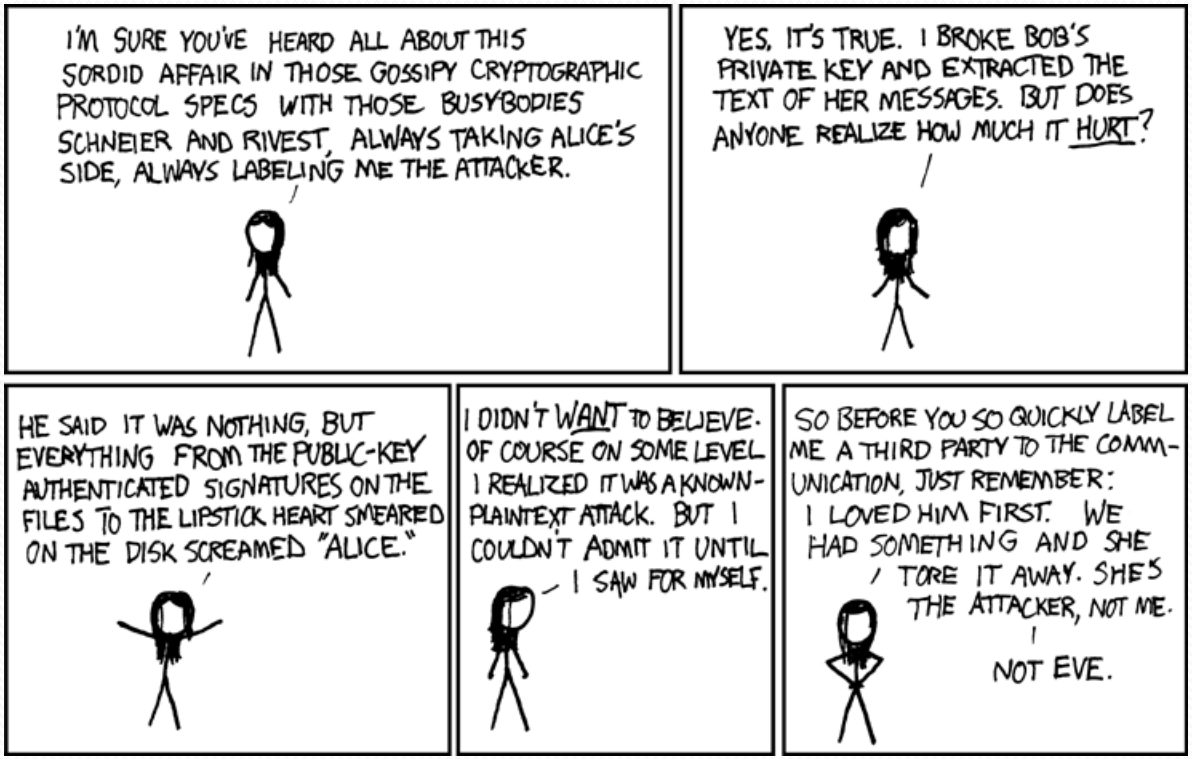
Nonrepudiation and confidentiality

- Nonrepudiation
  - Bob can't deny creating the signature
- If confidentiality needed on message
  - Sign, then encrypt message (or message + signature)
- Why not sign the encrypted data?
  - May inadvertently sign wrong info
    - Allows bait and switch by party sharing symmetric key
    - Would you sign an envelope containing a contract, hoping the contract is the one you think it is?
- Signing message then encrypting both is analogous to signing a contract, then putting it in an envelope

Signatures and encryption

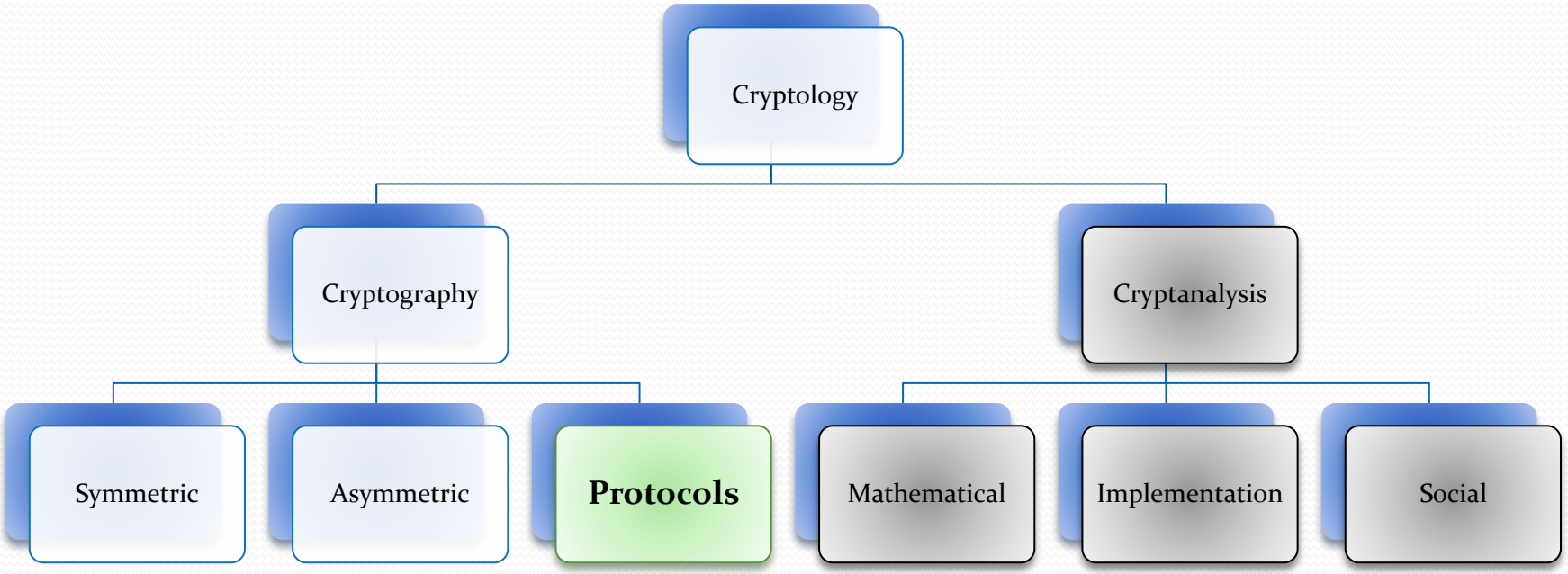
- Sign and encrypt message
- Sign and encrypt message + signature

- Why might one be better than the other?



http://xkcd.com/177/

Protocols

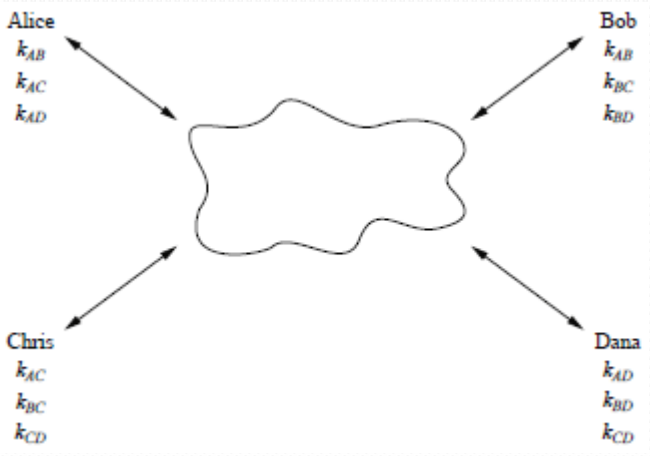

Cryptology
Cryptography
Cryptanalysis
Symmetric
Asymmetric
Protocols
Mathematical
Implementation
Social

Kinds of crypto protocols

- There are lots of crypto protocols
  - Key distribution
  - Multiparty computation
  - Zero-knowledge
  - Voting
  - And many more
- We'll focus on the most common: key distribution

Key distribution problem

- Assume every user in a network must share a symmetric key with every other user
  - N users means on the order of N^2 keys
    - Imagine that this network is the Internet
    - Management nightmare



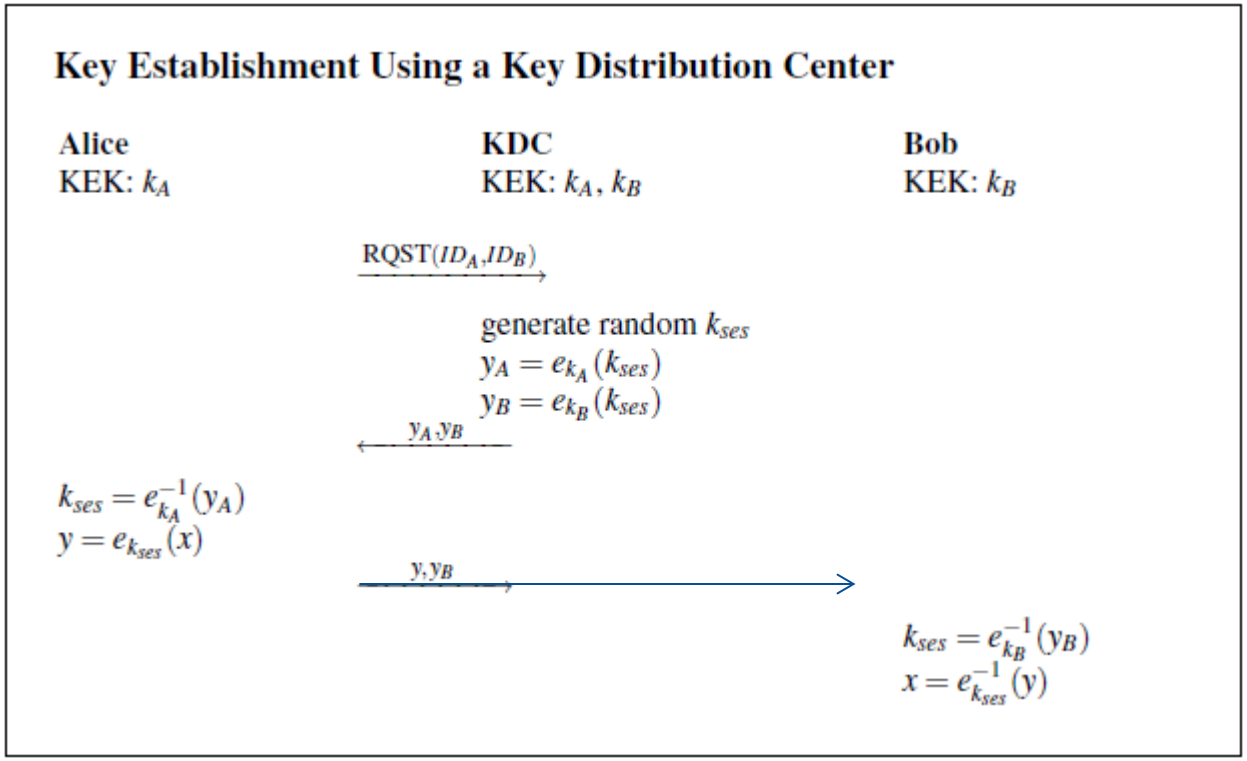
Key Agreement

- Key agreement is a very important concept in using cryptography
- Security of communications relies on secrecy of key
- How can parties exchange keys without compromising the key?
  - Parties create key together
  - One party creates key and sends to the other
  - Key derived by trusted third party and sent to both

Key distribution center

- Can use symmetric keys to create a session key
- Requires a trusted third party, KDC
- Alice and KDC share a key encrypting key (KEK)
  - Same for Bob
- Alice establishes key with Bob
  - Requests a key from KDC
  - KDC generates a short-term key
  - Distributes key to Alice and Bob, encrypted by respective KEKs
  - Alice and Bob now share a key

KDC example


Key Establishment Using a Key Distribution Center
Alice
KEK: k_A
KDC
KEK: k_A, k_B
Bob
KEK: k_B
RQST(ID_A,ID_B)
generate random k_{ses}
$y_A = e_{k_A}(k_{ses})$
$y_B = e_{k_B}(k_{ses})$
y_A,y_B
$k_{ses} = e_{k_A}^{-1}(y_A)$
$y = e_{k_{ses}}(x)$
y,y_B
$k_{ses} = e_{k_B}^{-1}(y_B)$
$x = e_{k_{ses}}^{-1}(y)$

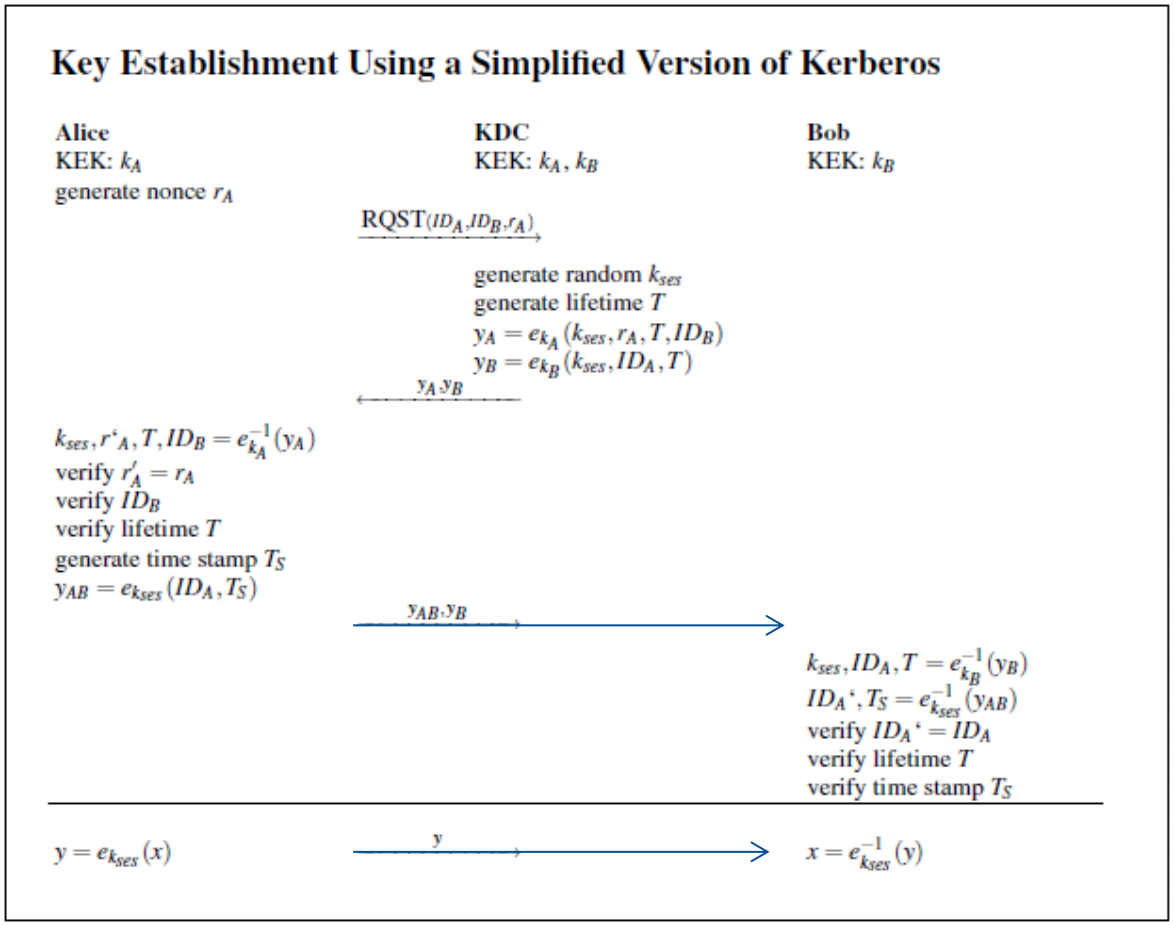
KDC pros and cons

- Pros
  - KDC stores N KEKs
  - Each user only needs KEK
  - Key establishment once for each new user
    - This secure channel outside scope of discussion
- Cons
  - If KEK compromised, adversary can masquerade as Alice
  - Replay
    - Adversary can impersonate KDC and send Alice and Bob old keys
    - No way to ensure freshness
  - Trusts that session key is with Bob, but may not be able to verify
    - Adversary can change request
  - Single point of failure
    - KDC compromise brings system to a halt

Kerberos

- Well-known KDC protocol
- Time matters
  - Session keys have lifetime
    - Lifetime encrypted
  - Prevents replay
- Challenge to KDC
  - Alice knows she's talking to KDC, not adversary

Simplified Kerberos



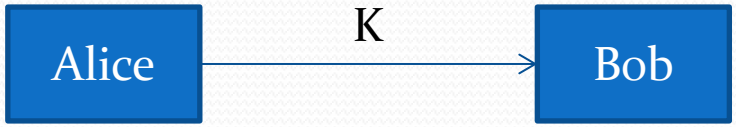
Cons

- Fixed some of the cons of first KDC protocol
- Remaining
  - Single point of failure
  - Lack of forward secrecy
    - If adversary compromises session key, all communications under that key are at risk
    - Only for duration of key lifetime
  - Setting up KEK for new user

Key establishment models

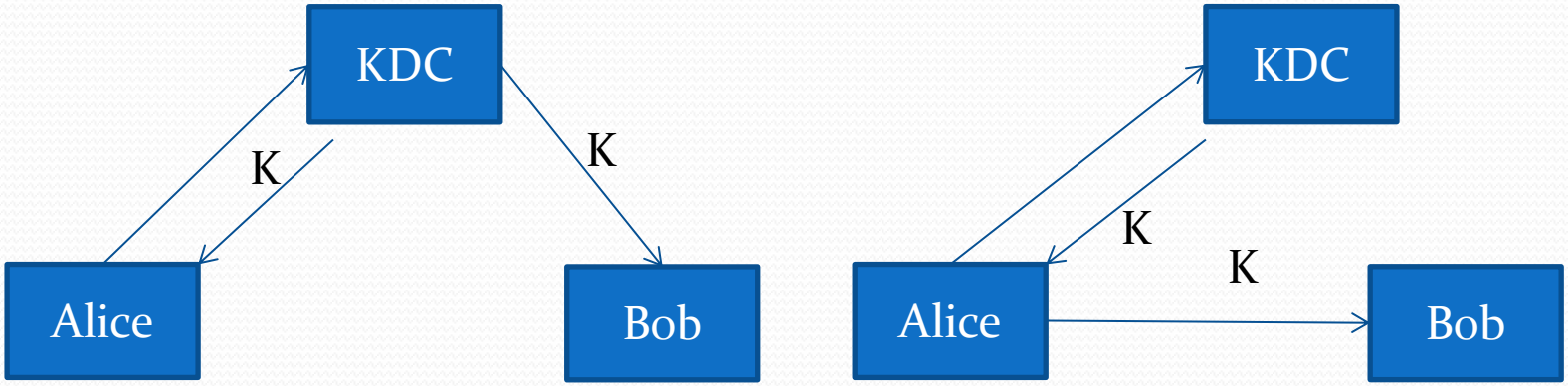
Point-to-point centralized

- One party gives the key to the other party
- Direct communication



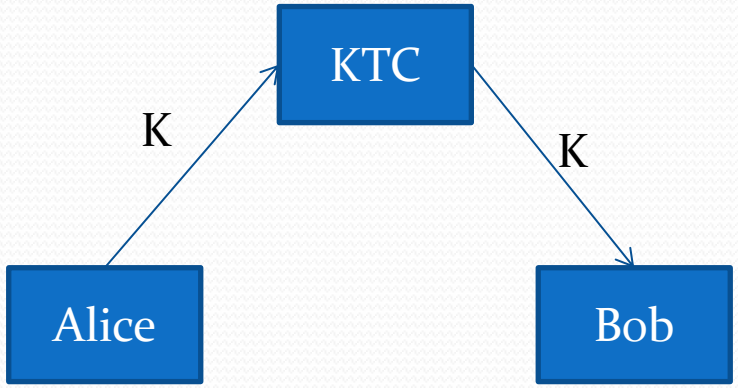
Centralized key management

- A single party generates and manages keys
  - May distribute to both parties
  - One party may be responsible for passing it on
- Each party shares a distinct key with the KDC, but not each other
- Centralized key generation



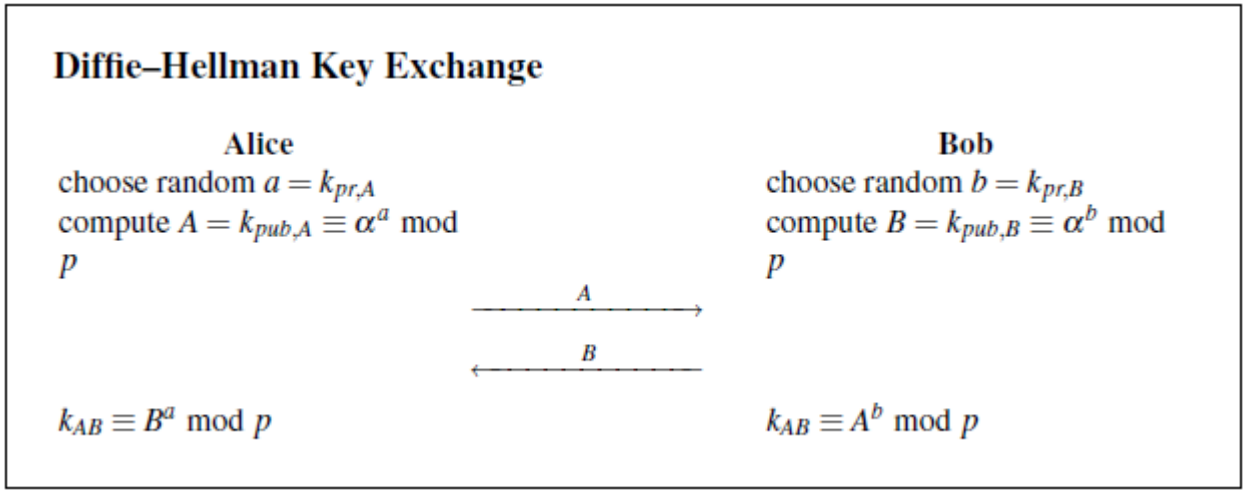
Key translation center

- Like centralized key management, but key generated by a party
- Distributed key generation



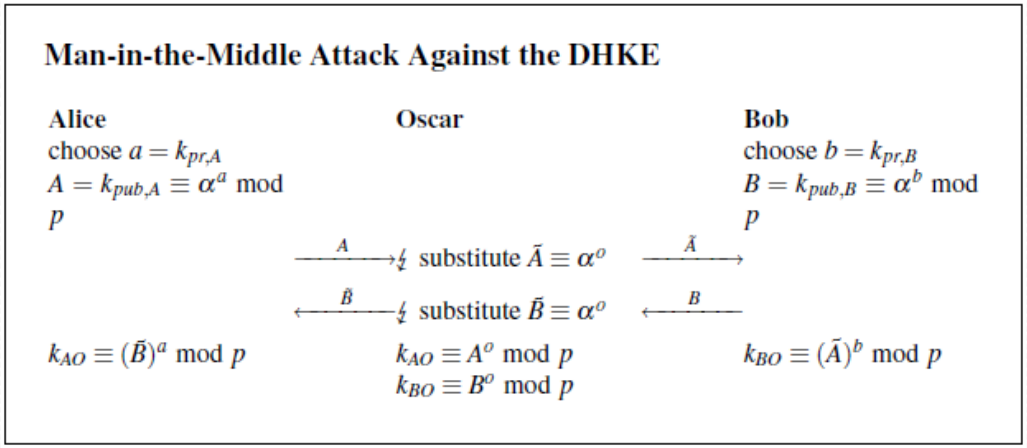
Diffie-Hellman key agreement

- Asymmetric method
- Both parties agree on parameters α and p
- Each party chooses a secret exponent



DH pros and cons

- Pros
  - Does not require a trusted third party
  - Does not require any other keys
- Cons
  - Does not authenticate either party (may not be a con)
  - Allows man-in-the-middle (MITM) attack



Certificates

- Provide binding between public key and party
  - Public key bound to Alice
  - Public key bound to Bob
- Alice and Bob can verify that they are talking to each other, not someone in the middle
- Generated by certificate authority (CA)
  - Trusted third party

Key transport with CAs

- Trusted authority issues certificates
  - Signed by trusted authority
- Key exchange with certificates
  - Alice gets Bob's certificate from Bob
  - Verify that Bob's certificate is valid using CA public key
  - Create session key
  - Encrypt session key with Bob's public key (in certificate)
  - Send to Bob
  - Bob decrypts with private key
  - Session key established

Obtaining keys

Certificate Generation with User-Provided Keys

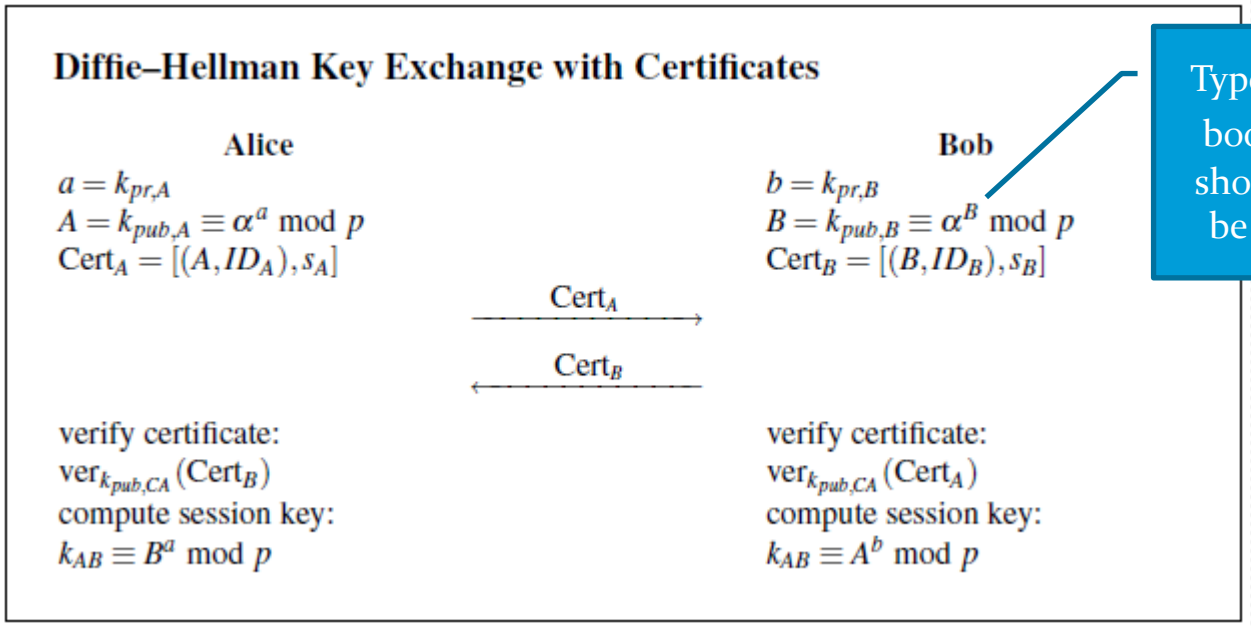
Alice		CA
generate $k_{pr,A}, k_{pub,A}$		
	$\xrightarrow{\text{RQST}(k_{pub,A}, ID_A)}$	
		verify ID_A
		$s_A = \text{sig}_{k_{pr,CA}}(k_{pub,A}, ID_A)$
		$\text{Cert}_A = [(k_{pub,A}, ID_A), s_A]$
	$\xleftarrow{\text{Cert}_A}$	

Certificate Generation with CA-Generated Keys

Alice		CA
request certificate	$\xrightarrow{\text{RQST}(ID_A)}$	
		verify ID_A
		generate $k_{pr,A}, k_{pub,A}$
		$s_A = \text{sig}_{k_{pr,CA}}(k_{pub,A}, ID_A)$
		$\text{Cert}_A = [(k_{pub,A}, ID_A), s_A]$
	$\xleftarrow{\text{Cert}_A, k_{pr,A}}$	

DH with certificates

- Certificates for authentication
  - Requires trusted certificate authority



Typo in book: should be α^b

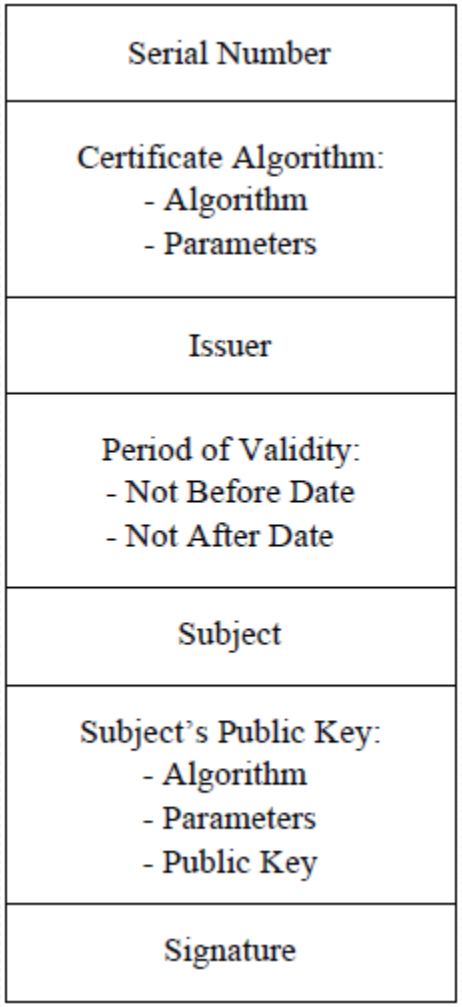
CAs and PKI

- Public-key infrastructure (PKI)
  - CAs + support
    - Key storage, distribution, etc.
- CA1 signs CA2, CA2 signs CA1
    - Parties with CA1 certificates can now talk to parties with CA2 certificates
- Ability to revoke certificates
  - Certificate Revocation List (CRL)
    - Blacklist
  - Alice checks CRL to verify that Bob's certificate is not on the list
  - If it is on the list, reject it

X.509

- Minimum certificate contains key and identity
- X.509 contains a lot more

- In IE, go to Tools > Internet Options > Content > Certificates
  - Browse certificates
- Go to your favorite login page (e.g. gmail)
  - Right-click on the page
  - Properties > certificates



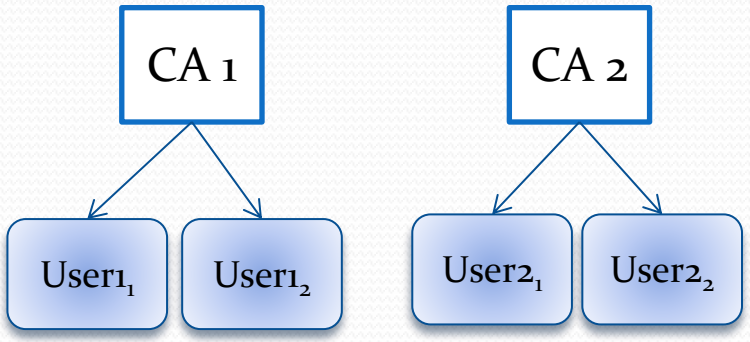
Exercise: certificates and RSA

- Most (if not all) the certificates just seen were RSA
- What might an RSA key exchange with certificates look like?

CA trust models

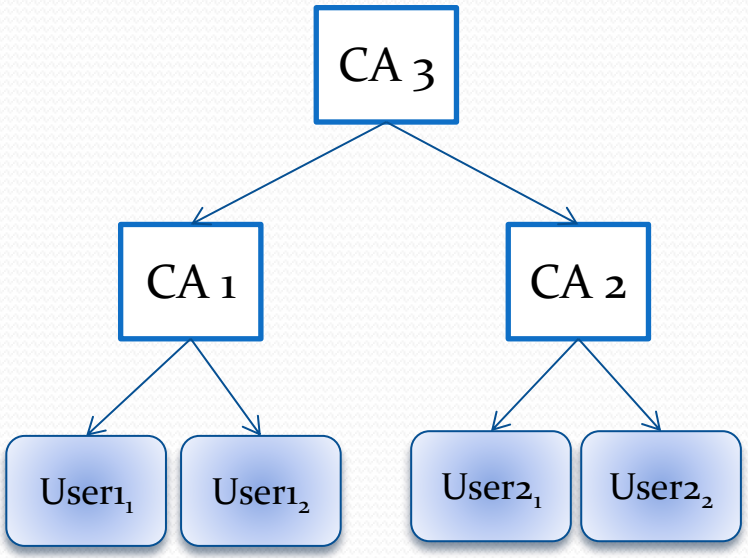
Trust with separate domains

- Two CA's have no trust relationship
- Each CA is part of its own domain
- Users in one domain unable to verify authenticity of certificates originating from other domain



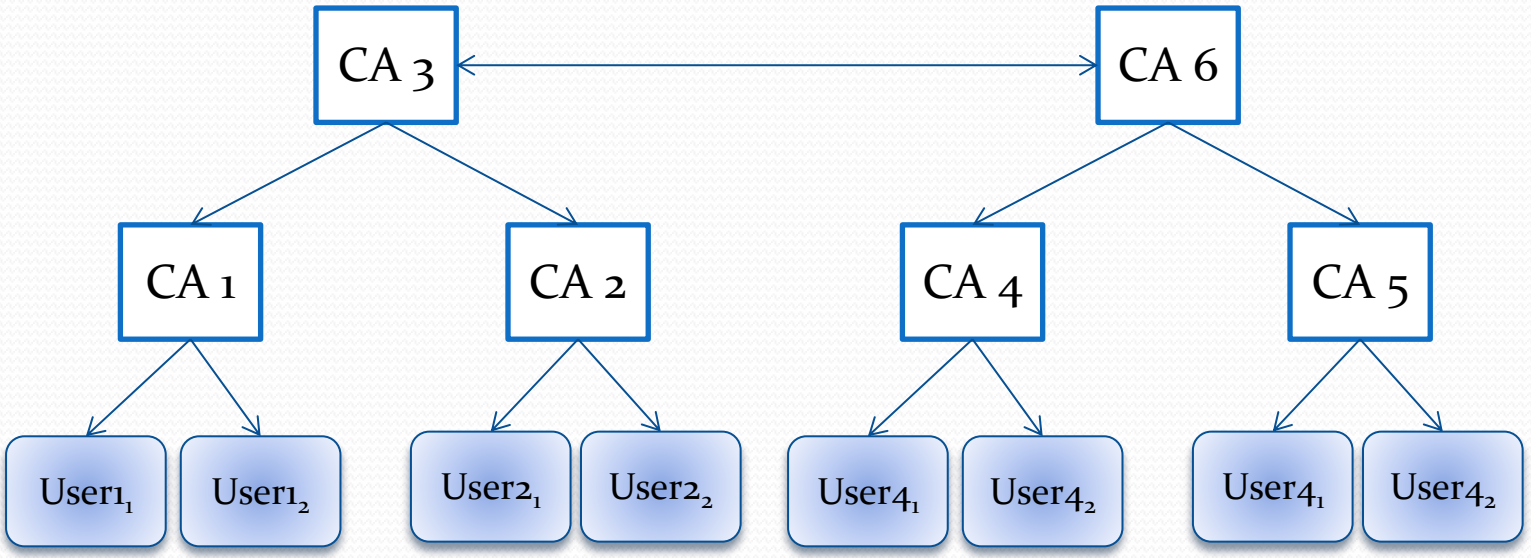
Strict hierarchical trust model

- Interoperability between domains
- Rooted tree
- Verify certificates all the way up to the root



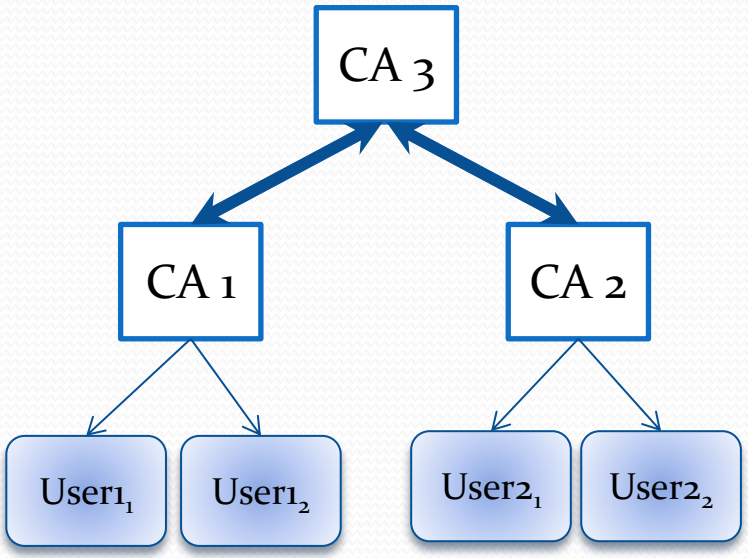
Multiple rooted tree

- Cross-certificate between two CAs
  - Certificate for X signed by Y
  - Certificate for Y signed by X



Reverse certificates

- CAs lower in the hierarchy also sign for CAs that are higher
- Forward certificate
  - Child certificate signed by immediate parent
- Reverse certificate
  - Parent certificate signed by immediate child

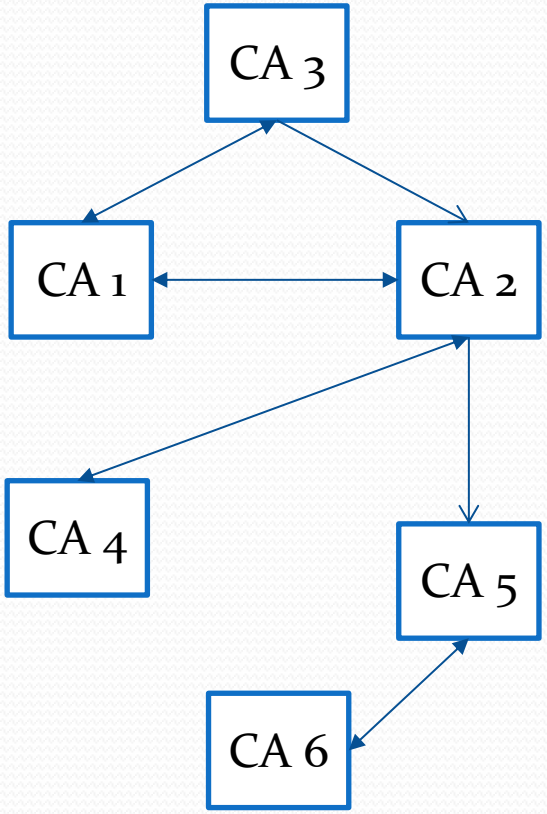


Reverse certificates (continued)

- Verification different than strict hierarchical model
- Start with public key of CA that created certificate
- Find least common ancestor
  - Do not need to go up to the root

Directed graph trust model

- Distributed trust model
  - Web of Trust
- No root or tree mandated
- An CA may cross-certify any other
- Each user entity start with its local CA

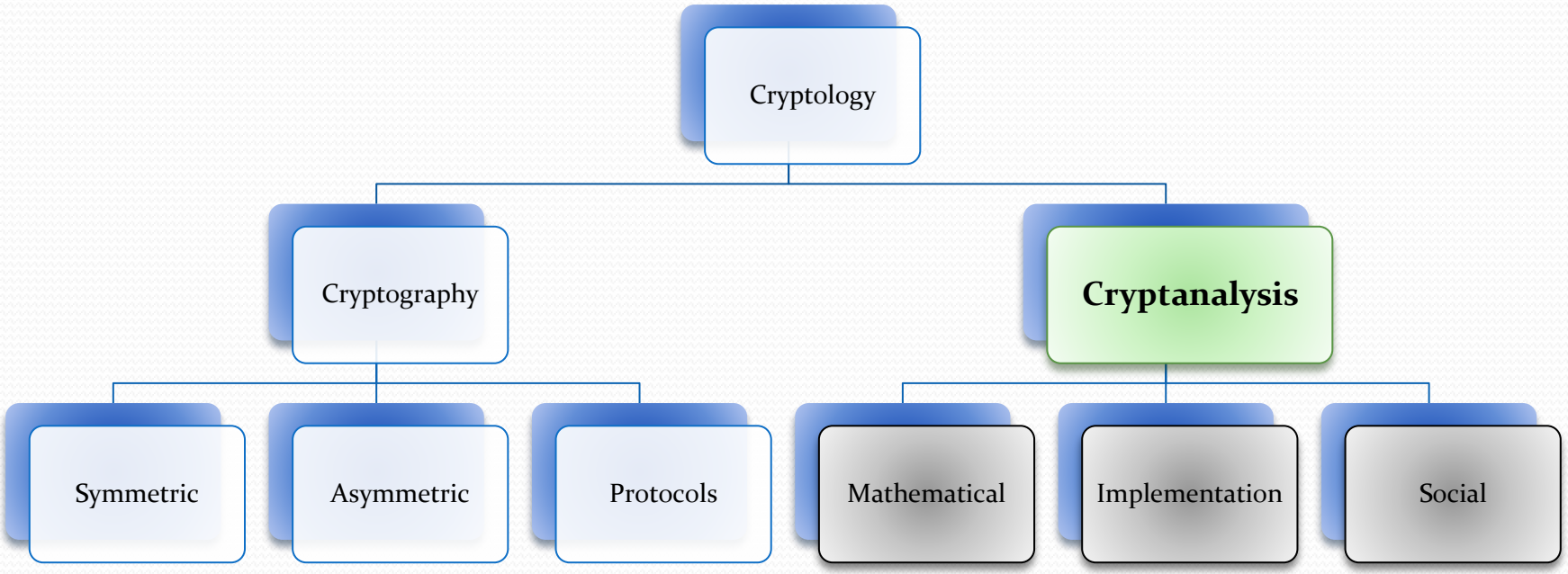


Food for thought

- Symmetric exchange
  - Keys have lifetime
  - Limit damage by adversary
- Asymmetric
  - CA revokes compromised certificates

- If you use biometric data as keys, what happens if you're compromised?
  - Gummy bears foil fingerprint readers
    - http://www.schneier.com/crypto-gram-0205.html#5
  - I'd like a new set of fingerprints, please

Cryptanalysis



Adversaries

- There are two kinds of adversaries to consider
  - Computationally unbounded
    - There is no limit to their computing power
    - Only OTP can beat them
  - Computationally bounded
    - Real world

Asking the Right Questions

- People sometimes ask me for crypto help

- Question: How do I break <insert current strong algorithm here>?
- My answer: wait 10 years

- Surprisingly, people don't like this
- Ok, I only give these kinds of answers to people I know pretty well
  - The reason I say this is because that is what they asked
  - But it is not what they *meant*

Attack models

- The adversary (who is sometimes you!) doesn't always have access to the same information
- Different levels of access enable different types of attacks
  - Each of these levels of access is captured by a model
- Knowing what model you're working with will help you assess your options

Brute force

- The most naïve approach
- Will always work
- Should take a long, long time
  - The message should no longer be sensitive by the time the attack succeeds

Ciphertext-only

- Goal: find key or plaintext
- Access to a set ciphertexts

- This applies whenever access to encrypted data is available
  - Confiscated computer with FDE
  - Capturing encrypted network traffic

Known plaintext

- Goal: find key
- Access to a set of plaintext-ciphertext pairs, created with the same encryption key

- This applies when there is knowledge of the plaintext the corresponds to captured ciphertext
  - Bad implementation leaves plaintext data structure
  - Data structure or packet payload always the same
  - Obtained plaintext through another channel

Chosen ciphertext

- Goal: find key or plaintext
- Access to decryption function, possibly resulting plaintexts
  - If errors occur, such as incorrect plaintext format, plaintexts may not be released to adversary
- Attack may be adaptive
  - Create ciphertext based on what you've learned
- Applicable when decryption requests can be made
  - Trusted computing
  - Anything that doesn't ask for the key for decryption

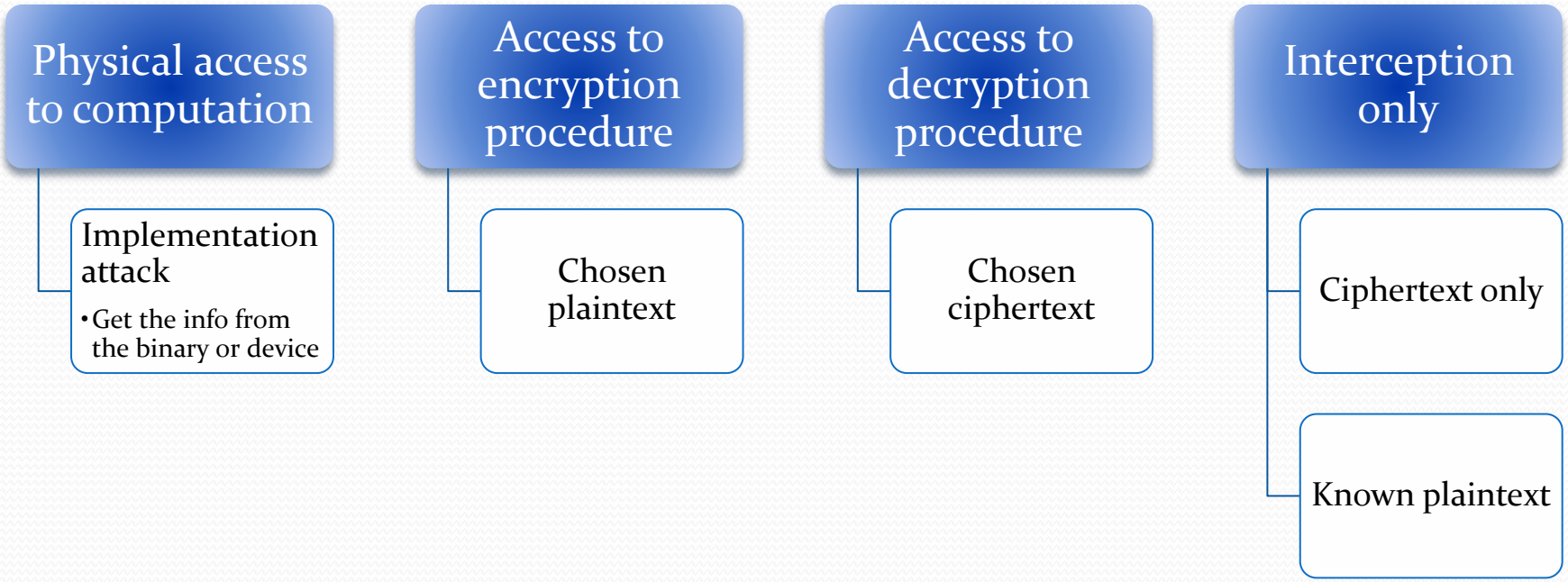
Chosen plaintext

- Goal: find key
- Access to encryption function and resulting ciphertexts
  - May be adaptive
- Applicable when encryption key not in adversary's control
  - Service offered by entity or device
  - Smart cards
  - Crypto chips

Other information

- Access to implementation
  - Binary data
  - Hardware module
- May be able to get data from the implementation without going after the crypto itself

Good bets

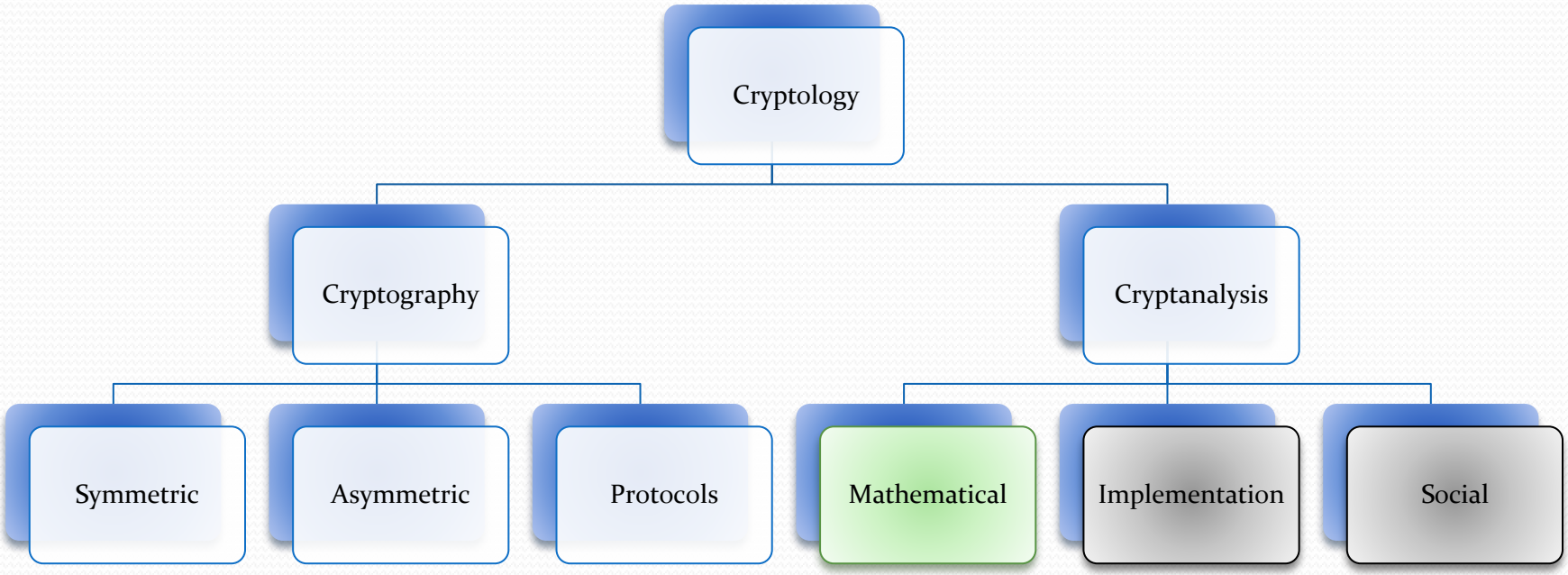

Physical access to computation
Implementation attack
•Get the info from the binary or device
Access to encryption procedure
Chosen plaintext
Access to decryption procedure
Chosen ciphertext
Interception only
Ciphertext only
Known plaintext

And more

- Sometimes there is more info to leverage
- Use in conjunction with one of these models

- Example
  - Related-key
    - Known-plaintext
    - Chosen ciphertext
  - Use multiple keys instead of single key
    - Know something about the relationship between them

Mathematical (analytical) attacks

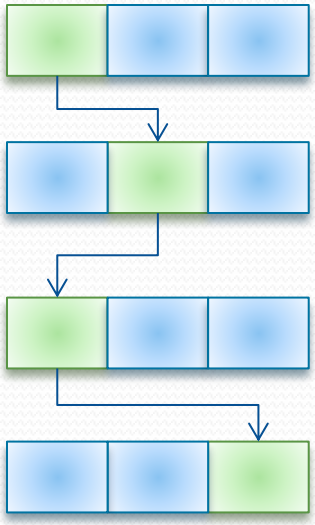


Of more theoretical interest

- There are several analytical attacks out there
  - Lots of beautiful math
- Symmetric cornerstones
  - Linear cryptanalysis
  - Differential cryptanalysis
- Asymmetric
  - Factoring
  - Discrete log
  - Tricks when bad parameters used to make key

General principles for analytic attacks

- Reduce your work
  - Isolate parts of the state/key as much as possible
  - $2^{n/3} + 2^{n/3} + 2^{n/3}$ is a lot smaller than 2^n
    - Example: n=9
      - 8+8+8 = 24 vs. 512
    - This is why diffusion is important
      - Prevents adversary from isolating sections



Symmetric analytical attacks in brief

- Linear cryptanalysis
  - Find linear relationships between plaintext, ciphertext, and key
  - Allows adversary to find parity of the key
  - Reduces key space search considerably
- Differential cryptanalysis
  - Known difference between plaintexts results in certain ciphertext difference with some probability
  - Difference could be introduced into ciphertext instead of plaintext
  - Difference could be in related keys

Linear vs. differential

- Which one requires a stronger adversary?
- What models might each use?

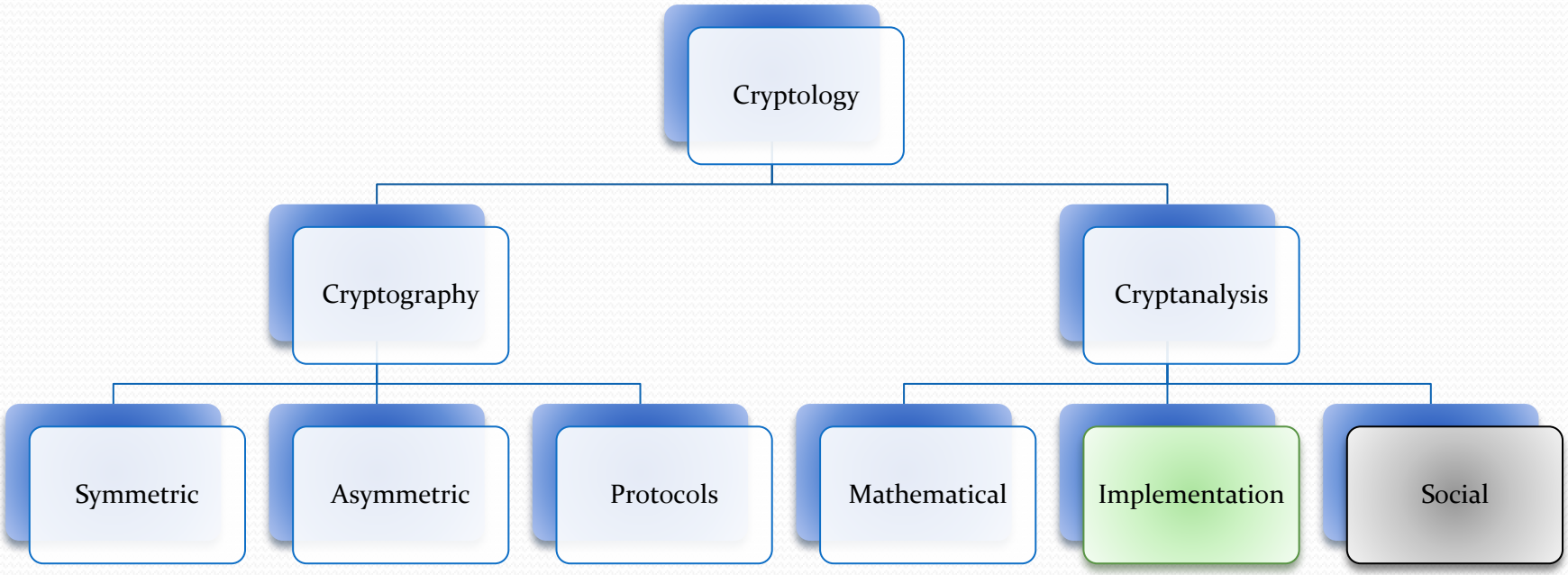
Factoring

- Suppose you don't have a quantum computer
- You can factor 512-bits on your own with publicly available tools
  - Msieve, GGNFS
- Currently, record is 768-bit integer
- Took 3+ years by experts
- General number field sieve (GNFS)
  - Relies on heuristics and a lot of memory and computing power

Discrete log

- Here's a high level view of two methods
- Index calculus method
  - Probabilistic algorithm
  - Linear algebra
  - Group theory
- Pollard's rho algorithm
  - Generator generates all values of the finite set
    - There must be a cycle
  - Find a cycle
    - x_i congruent x_{2i} modulo m
    - Gives equation that can be used solve discrete log directly
  - Has analog for factoring

Implementation attacks



Implementation Matters

- A mathematical specification is one thing, but an implementation is a whole new beast
- Implementations leak information
  - Vulnerable to side channel cryptanalysis
- Examples
  - Time to compute
    - Different inputs to an operation may cause subtle timing differences
      - Older OpenSSL implementation was subject to this
      - Cache attacks on AES cloud computing environment

Power and EM attacks

- Different operations take different amounts of power
  - Simple power analysis (SPA)
  - Differential power analysis (DPA)
- Operations alter electromagnetic field differently

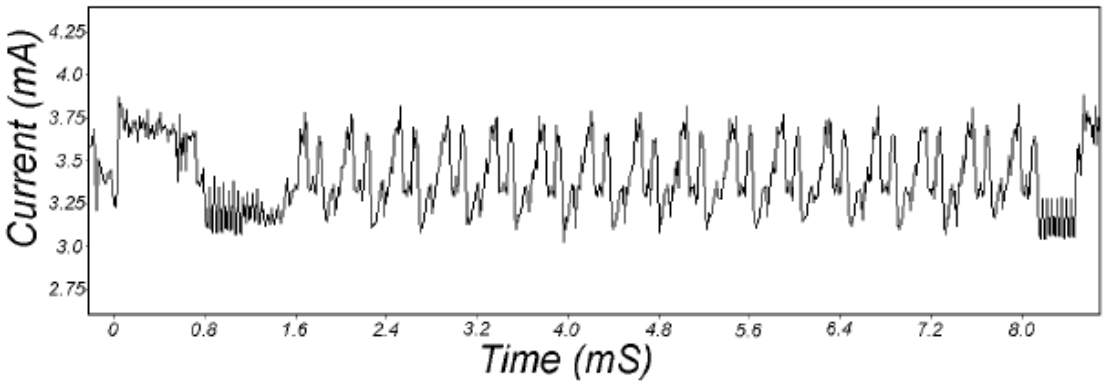


Figure 1: SPA trace showing an entire DES operation.

Image from "Differential Power Analysis" by Kocher et al.

Power and EM fields (continued)

- Can perform adaptive chosen ciphertext attacks to obtain plaintext and key, a small amount at a time
- Can perform adaptive chosen plaintext to find key

Power and EM: an example

- Suppose guessing bit correctly makes a spike in a particular location
- Start with key all zeros, key is n bits
- For i=0 to n-1
  - Decrypt or encrypt with
    - k[i]=1
    - k[i]=0
  - Keep the value that causes a bigger spike
- O(n) encryption/decryption

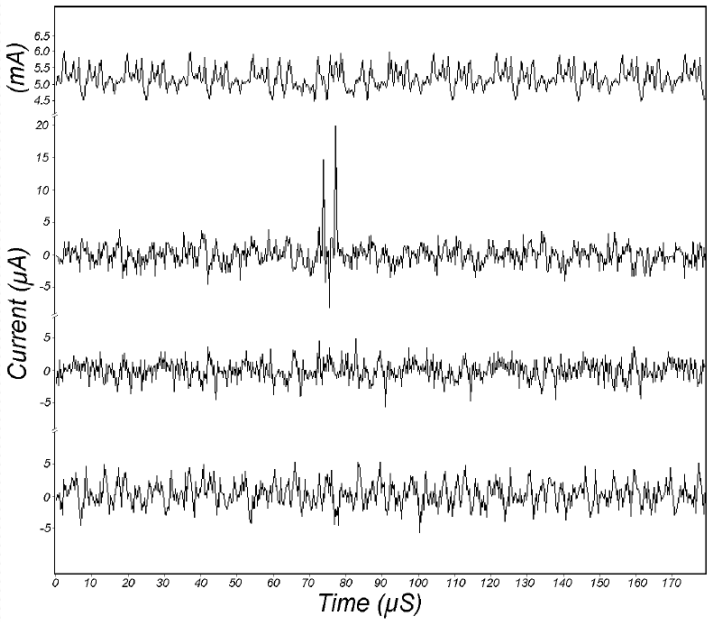


Figure 4: DPA traces, one correct and two incorrect, with power reference.

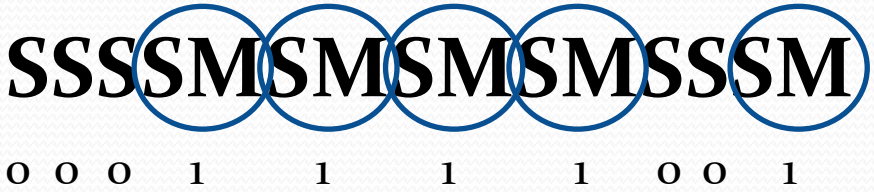
Grabbing the key

- The key has to be exposed somewhere
- Software implementation
  - In memory
- Hardware
  - In flash memory or hard coded

- Physically extracting key from hardware is tricky
  - Need to shave off chip to get at gates or memory
  - May destroy chip
    - May accidentally destroy key

Example: square-and-multiply side channel

- An efficient way of computing $x^e \bmod n$ is using an algorithm called *square-and-multiply*
- Starts with $a=1$
- Work from most significant bit of e down
  - If current key bit is 1, multiply follows square (SM)
    - $a=a^2$
    - $a=ax$
  - If current key bit is 0, square alone (S)
    - $a=a^2$
- What is the value of e based on this string?



Exercise: securely computing square-and-multiply

- Break into groups and answer the following
  - What is it that makes the attack on the previous slide possible?
  - How can you prevent this attack on square-and-multiply?

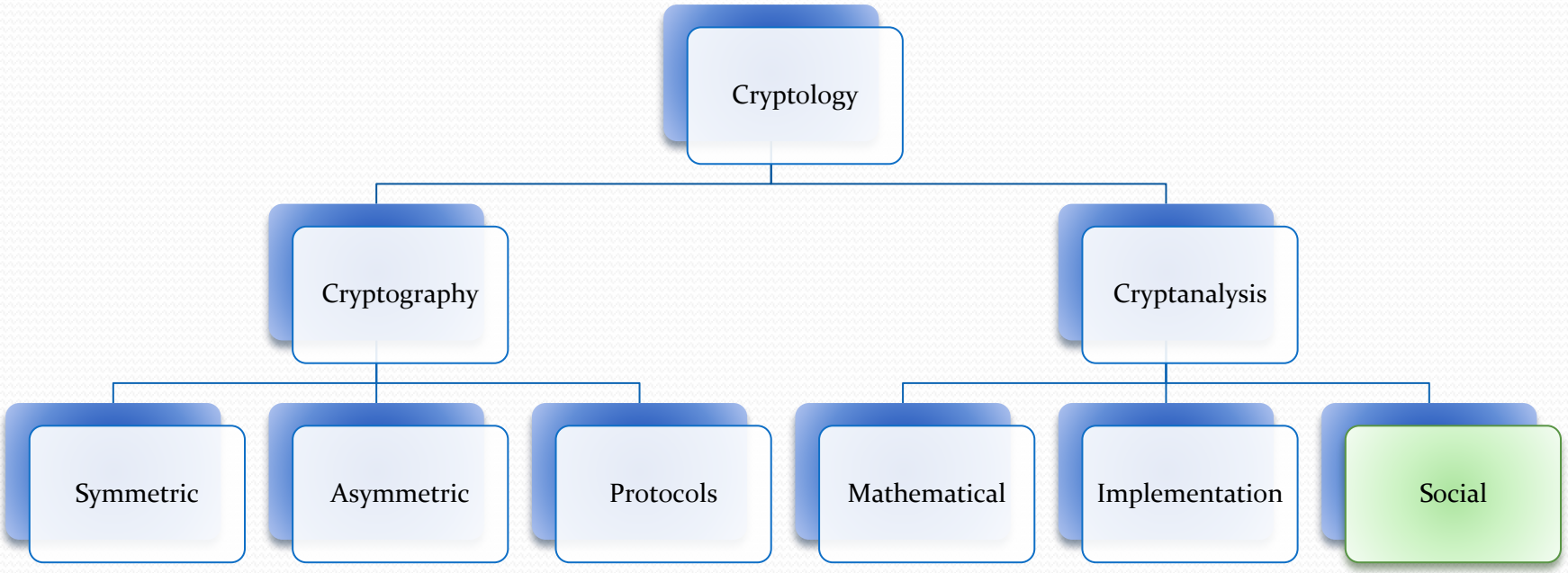
Demo

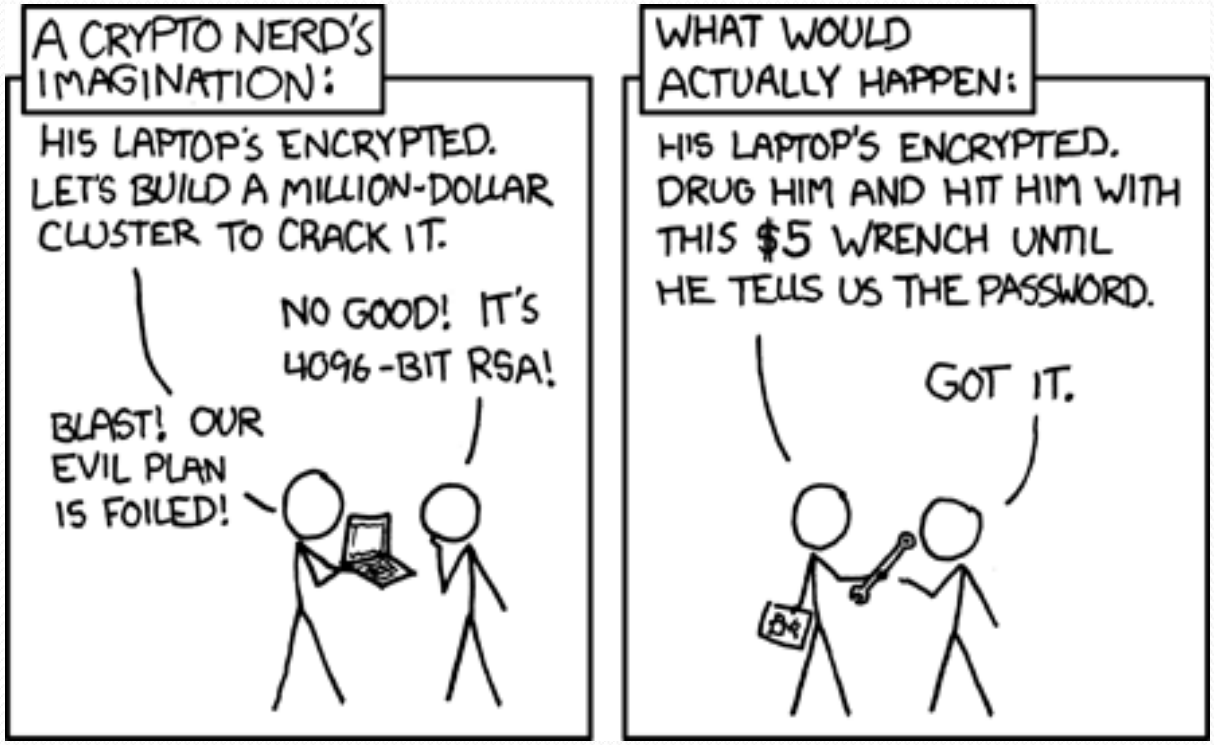
- What do square and multiply operations look like?
- JCrypTool demo

Attacks: further reading

- “Remote timing attacks are practical” by Boneh and Brumley
- “Differential Power Analysis” by Kocher et al.

Social Attacks





http://xkcd.com/538/

Social attacks

- Attacking the crypto is usually pretty hard and requires skill (unless an attack has been included in a tool)
  - Why not just go around the crypto?
- Humans are generally the weakest link in a system
- People can be coerced
  - Someone holds a gun to your head and says "your secret key or your life"
    - What would you do?
  - What if they threatened someone you love?
- Espionage
- Blackmail
- Bribery

Phishing and impersonation

- Your account has been suspended due to suspicious activity. Please follow the link and enter your username and password to verify your identity and reactivate your account.
  - Completely bypasses the cryptography

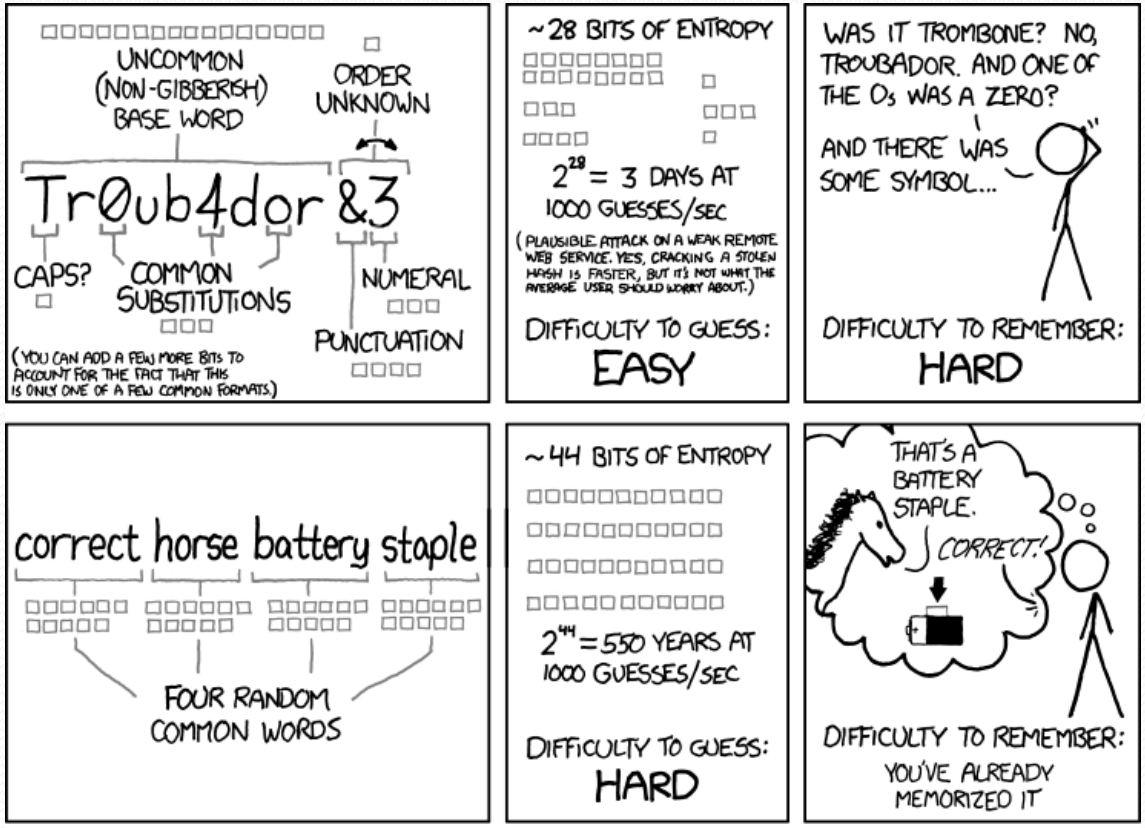
Policy attacks

- In addition to social attacks on people, can attack policies
- Policies force certain behavior
  - Passwords
    - Common to derive secret key from password
  - Backward compatibility
- Policy knowledge can give the adversary an advantage
  - Password rules
  - Force use of deprecated cipher

The human brain & passwords

- People can remember 5-9 chunks of information
  - Unfamiliar things make smaller chunks
    - E.g. “Rks9%3”
    - Each character a chunk
  - Familiar things enable larger chunks
    - E.g. “four score and seven years ago”
    - The entire phrase one chunk (or 6 chunks)
- “four” comes from a much larger set than “R”
  - Words in phrase related, so phrase not high entropy
- “four pig ninja gumby minitrue fire” less predictable

Password rules



http://xkcd.com/936/

Exercise: identifying attack vectors

- Imagine a protocol where a button is pressed on a remote, and a light turns on or off
- Remote is physical device, i.e. dongle
- The transmitter and receiver each have a shared fixed value, v, and a counter, ctr
  - Counters increment and are synched
- Transmitter and receiver share a symmetric key, K
- Transmitter sends encrypted message E(v||ctr, K) to receiver
- Receiver decrypts message
  - If v is correct and ctr is in acceptable range, turn on/off light and increment counter
  - Else, do nothing

Exercise: identifying attack vectors (continued)

- Your goal: turn the light on/off
- What are possible attack vectors?

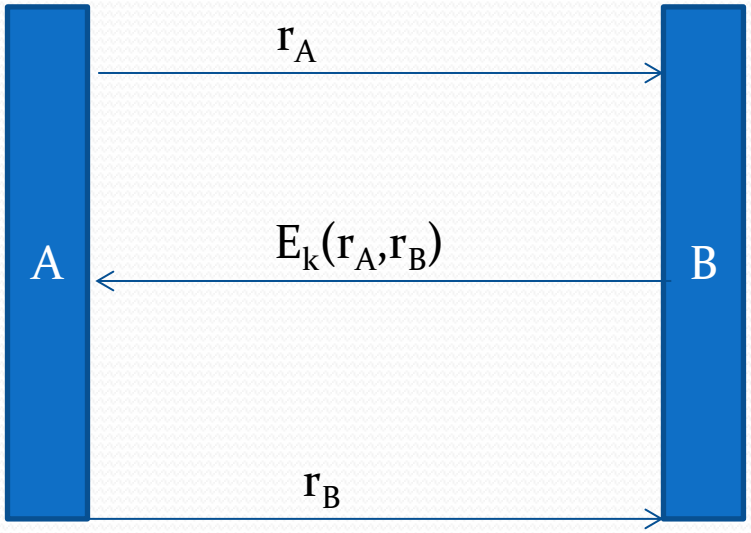
How to attack a protocol

Man-in-the-middle

- Saw this with Diffie-Hellman
- Convince A and B that they are talking to each other
- In reality,
  - A is talking to E (who claims to be B)
  - B is talking to E (who claims to be A)
  - E is transparent to both A and B

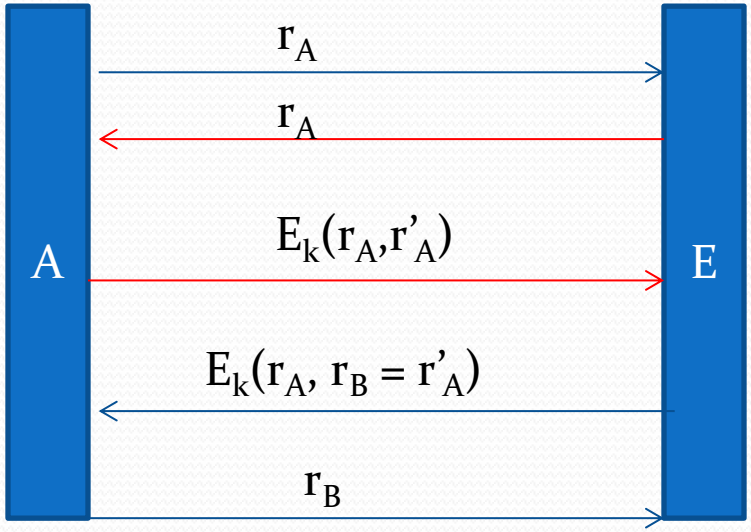
Reflection

- Suppose A and B share a key, k
- Suppose they authenticate each other with the following protocol



Reflection (continued)

- E can impersonate B by reflecting messages back to A
  - Two protocols running at once
- Intercept messages
- A believes it has authenticated B, but B was not involved at all

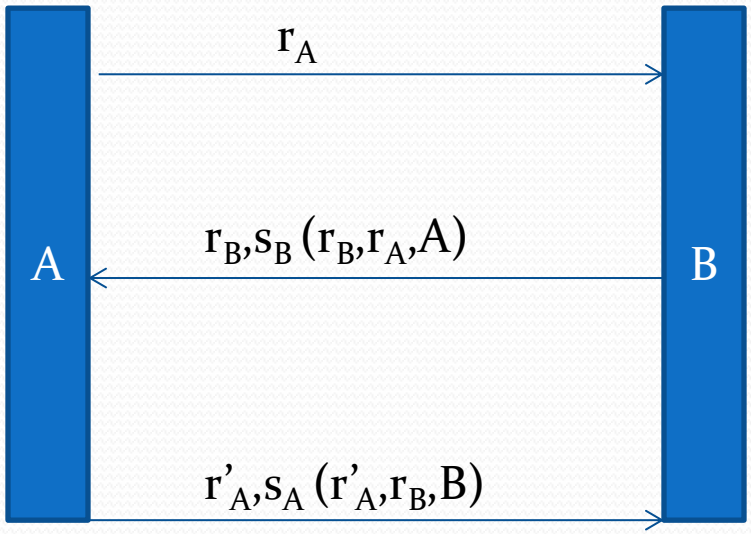


Mitigating reflection

- Different keys for different communication directions
- Include identifier of originating party in messages
  - If A gets an encrypted message from A, then reflection can be detected

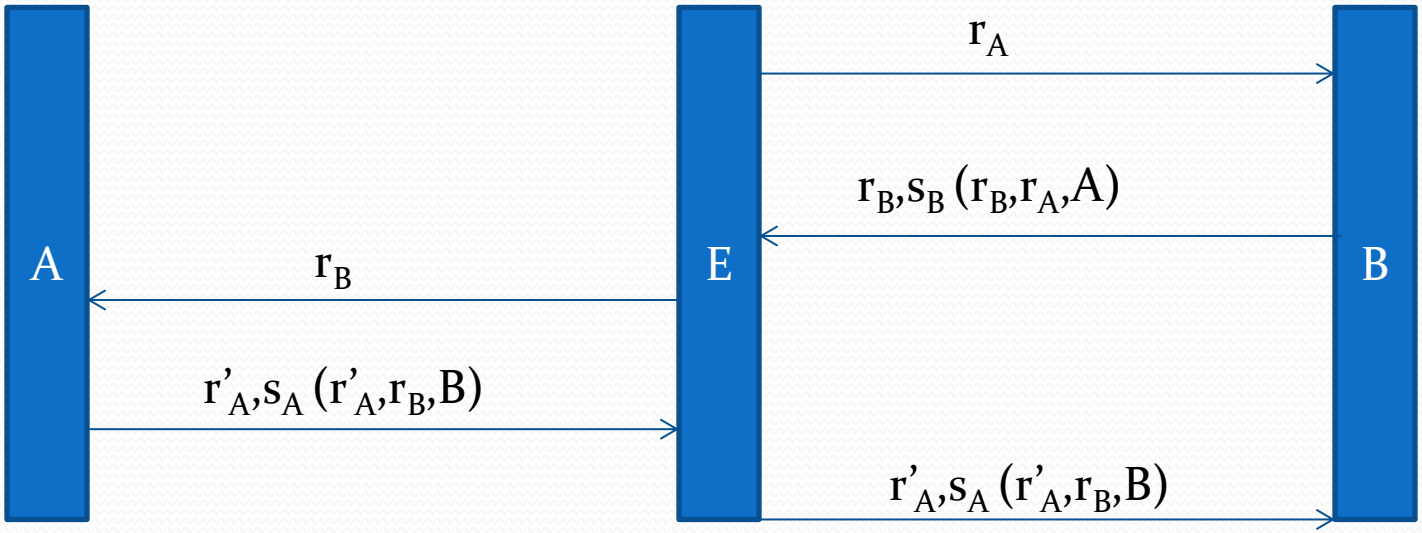
Interleaving attack

- Assume all public keys authentic
- A and B choose random values and sign them



Interleaving attack (continued)

- Authenticate with B as A
  - E initiates with B as A
  - E initiates as A with B
  - Use messages from one exchange to finish the other



Misplaced trust

- A trusted third party is inherently trusted
- What could happen if the TTP is ***not*** trustworthy?

When is an algorithm broken?

Is it broken?

- Think of your data (plaintext or keys) as delicious candy and an encryption algorithm as a piñata
  - That makes cryptanalysts kids with bats
- They keep hitting the piñata, making dents
- Eventually a seam splits
- Then it cracks open and the candy falls out
  - It is clearly broken at this point
- The dents don't mean its broken, but they start to add up
  - You may even be able to get a couple pieces of candy out of a split seam

Is it broken? (continued)

- Attacks accumulate
- An "academic" attack hits the scene
  - Improved over many iterations
  - Eventually becomes practical
- If best practical attack has complexity below security threshold, the algorithm is broken
- If there's not much left, it is close enough and you should already be transitioning
- If a property you were counting on is compromised, transition now

- Example: new attack on AES shaves off almost 2 bits
  - Even with that, 126 > 112, so it is ok for now

Maybe its broken, maybe its not

- All algorithms will eventually be broken
  - It is a matter of when
- You should start to worry when the attacks look like they might become practical soon
- Example: SHA1
  - 2004: Attacks in SHA-0
    - Plans to attack SHA-1
  - 2005: Attack on SHA-1
    - Collisions, not preimages
    - Not recommended in new applications
    - Ok in legacy applications, where collision less important
  - Today: SHA-1 still in use (have > 2^{50} complexity for collision)
    - SHA-2 family preferred
    - SHA-3 standard in progress

It will all break eventually

- There is often a significant cost to changing algorithms
  - Especially in hardware
- Backward compatibility is always an issue
- Cryptology is dynamic, not static
  - Attacks take a long time to mature, but break could be tomorrow
  - When algorithm crippled, need to act quickly
  - Plan for change in any design

Case Studies

When good crypto goes bad

Important

- There is crypto, and there's the system it is part of
- As soon as you look at crypto as part of a system rather than a stand-alone object, things change
- You need to be mindful of the attack vectors that you introduce

WEP

- Wired Equivalent Privacy
  - Secure wireless network communications
- WEP is a good example of doing it wrong
- But why?

WEP (continued)

- Uses RC4 stream cipher
- Seed has two parts
  - IV: 24 bits (public)
  - Key: 40 bits (secret)
- CRC32 for integrity

- Let's take a look at each of these

WEP: key and IV

- Export restrictions limited key size
  - 64-bit WEP is 40-bit secret plus 24-bit IV
- There is a 128-bit version
  - 104-bit secret plus 24-bit IV
- Cipher key = IV||secret

- First rule of stream ciphers
  - Don't reuse the same key stream
- 2^{24} is not large
  - Especially on a high-volume network
- Small IV space causes high probability of using same key stream more than once
  - BAD!

WEP: RC4

- Bias in key stream
  - Bytes do not occur with equal probability
  - First couple bytes have strong bias
  - For each packet, you're at the beginning of the key stream
    - Bias in every packet
- Attacker can collect several messages
  - Already knows IVs
  - Can get info on XOR of two messages
  - The secret is revealed with an obtainable amount of data and some analysis
- Note this is all possible from nothing but ciphertexts

WEP: integrity check

- CRC32
  - Cyclic redundancy check, 32-bits
- Linear and predictable
- Good for detecting random transmission errors
- Not cryptographically secure
  - Adversary can make changes such that CRC32(modified) = CRC32(original)
    - i.e. bit flipping
- Terrible integrity check
  - It wasn't designed for this purpose

WEP is good for something

- In all fairness, all ciphers broken eventually
  - WEP fails even if RC4 still worked well
- It is important because it does contribute to our knowledge of what not to do
- It also gives support to an attack model

- Related-key attack model
  - Some people say that it is unrealistic
  - People who defend it just need to point to WEP

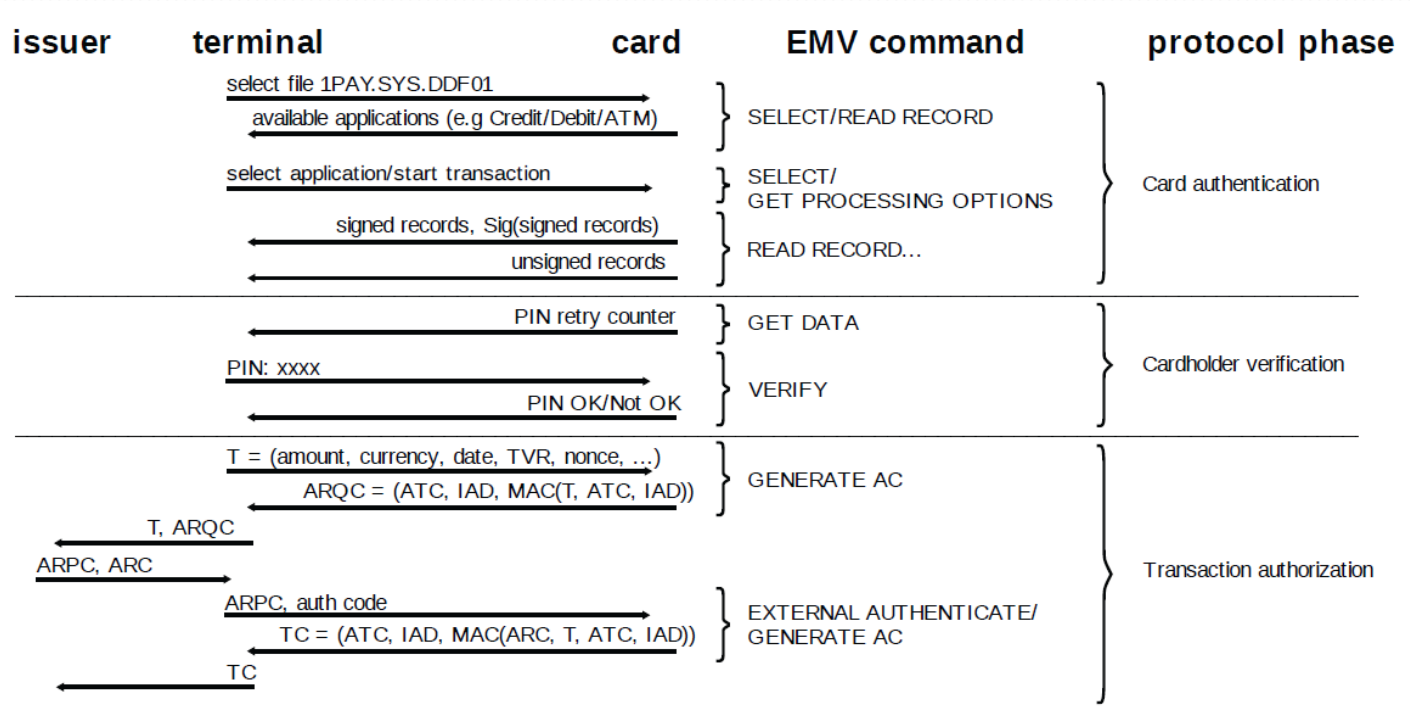
EMV Payment

- "Chip and PIN is Broken" by Murdoch et al.
- Europay, MasterCard, Visa
  - Prominent payment protocol
- Known as "chip and PIN"
  - Payer uses smartcard and PIN
- Goal: minimize cost of disputes from issuing bank

EMV Payment (continued)

- Two places for burden
  - Payer used chip and PIN
    - Verified, so user must be authentic
    - Burden on account holder
  - Payer signed
    - Merchant verified signature
    - Burden on merchant

EMV: protocol



- TVR: terminal verification results, ARQC: authorization request cryptogram, ATC: application transaction counter, IAD: issuer application data, ARPC: authorization response cryptogram, ARC: authorization response code, TC: transaction certificate

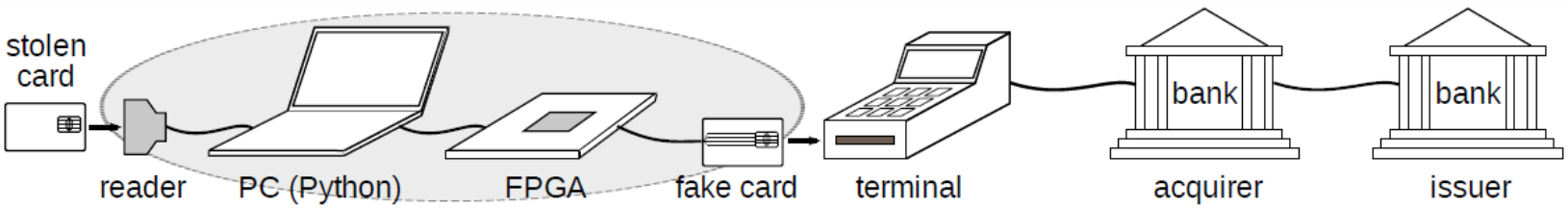
EMV: what went wrong

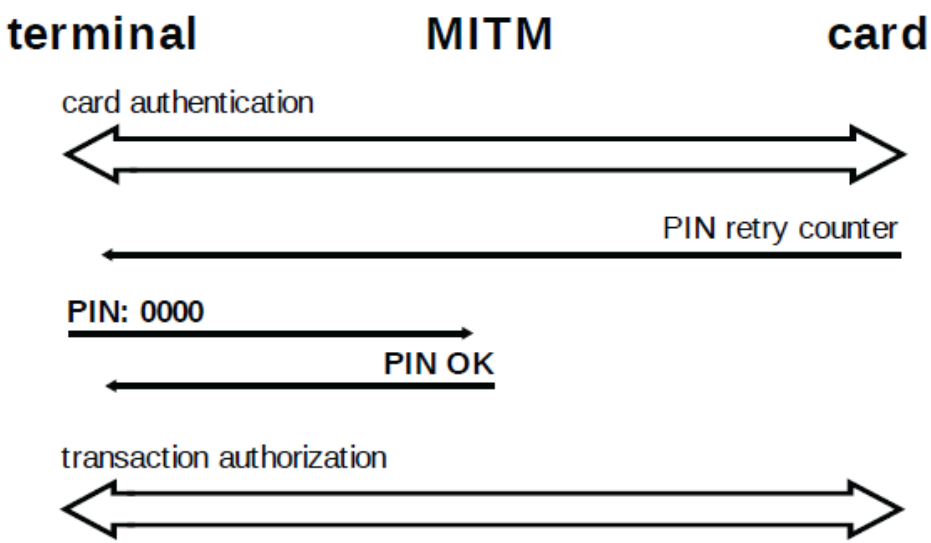
- It is possible to enter an unintended state
- Three parts of protocol are separated
  - Card authentication
  - Cardholder verification
  - Transaction authorization
- Partial views paint different views of the world
  - TVR doesn't specify what verification method used on success
    - Mainly lists failures
- Compartmentalizing lets adversary build their own verification process

EMV: attack

- Use FPGA, software, and dummy card to intercept communications with stolen card
- Alter cardholder verification
- Trick terminal into thinking PIN verification succeeded
  - Real card believes terminal does not support PIN verification
    - Thinks merchant verified signature
  - Merchant thinks PIN was verified
- Protocol records don't indicate how cardholder verification performed
  - No record to identify inconsistency

EMV: attack (continued)





EMV: attack (continued)

- Caveats
  - Requires hardware for man-in-the-middle
  - That wiring up your sleeve might raise suspicion on a hot summer day
- Authors of this paper have assisted in dispute cases where card stolen
  - Lucky customers have the encoded cardholder verification on receipts
    - Decoding shows that signature was used, not PIN
    - Were able to get money back
  - Unlucky customers (most of them) didn't have this record
    - No way to show signature use
- System does not provide adequate evidence

VOIP

- Phoneme - basic building block of speech
- Some VOIP protocols compress phonemes, then encrypt with stream cipher
  - Stream cipher encryption preserves length of message
  - Phonemes compress to different sizes

- What does this mean for the adversary?

VOIP: problems

- Correlation between packet length and phoneme
- Packet length gives information to attacker
  - Attacker can learn something about encrypted VOIP conversations
    - Who is talking
    - What language
    - What they are saying
- See “Uncovering Spoken Phrases in Encrypted Voice over IP Conversations” by Wright et al.

KeeLoq

- Remember the exercise where the remote turns on the light?
  - That was a simplification of KeeLoq code hopping scheme
  - Counter prevents replays
- Instead of turning on a light, it opens your car or garage door

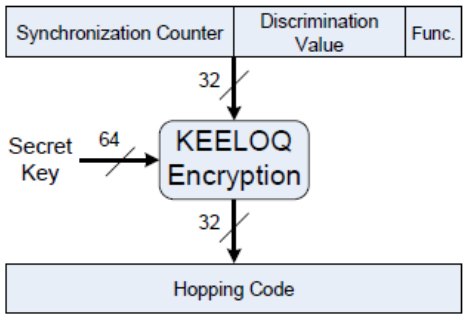
- Attacker's goal: open the door

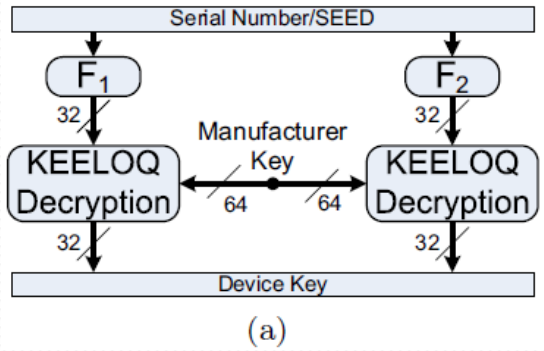
KeeLoq: attacks

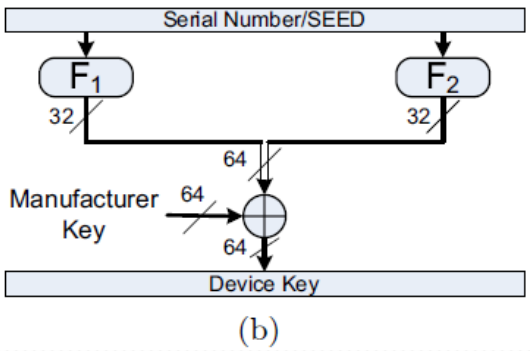
- Social attacks
  - May gain physical access
    - Clone dongle
  - Noticeable
- Can watch signals via side channel
  - No access to plaintext

KeeLoq: device key

- Device key is secret shared between receiver and transmitter
- Derived using a manufacturer key
  - Honda, Toyota, etc.
  - Once you know the manufacturer key, you can spoof anything you have the serial number for







KeeLoq: power analysis

- Differential power analysis used to find keys
- Cloning transmitter
  - With physical access to remote
    - 10-30 traces
  - Without physical access
    - Cloned by eavesdropping, if manufacturer key known
  - With device key, decrypt message to obtain fixed values and counter
- Denial of service
  - Increment counter so that “real” remote counter is outside acceptable window

KeeLoq: final notes

- Details in "On the Power of Power Analysis in the Real World: A Complete Break of the KeeLoq Code Hopping Scheme"
  - Defend against attacks with sufficiently random longer seeds
- The authors found manufacturer keys
  - To my knowledge, have only released them to the manufacturers as proof
  - Only need to eavesdrop at most two message to clone a transmitter
- It takes skill to pull off key extraction
  - Only need to use skill once to find manufacturer key
  - Rest can be performed by unskilled adversary

Standards

Standards

- Several international standards
- Some countries have their own standards
  - Some design internally
  - Some use designs from other countries
  - US has mix of NSA and internationally-designed crypto standards
    - DES, SHA-1 examples of NSA
    - AES, SHA-3 examples of international
  - Anyone who wants to do business with the US government needs to implement our standards
  - China likes algorithms designed in China
    - Needed their own stream cipher for 3G LTE (long term evolution)

Organizations

- This is not an exhaustive list

- Government organizations
  - i.e. National Institute of Standards and Technology (NIST), NSA
  - CRYPTREC (Japan)
- RSA Labs
- Internet Engineer Task Force (IETF)
- Institute of Electrical and Electronic Engineers (IEEE)
- American National Standards Institute (ANSI)
- ECRYPT II

NIST

- Computer security division provides a suite of approved cryptographic algorithms
- Block ciphers
  - AES (advanced encryption standard)
  - Triple DES (also called 3DES, TDEA)
  - DES (flawed, deprecated)
- Hash functions
  - SHA (flawed, deprecated)
  - SHA-1 (phase-out)
  - SHA-2 family
  - SHA-3 family

PKCS

- Public key cryptography standards
- RSA labs
- RSA standards (PKCS #1)
  - Encryption and signatures
- Diffie-Hellman key agreement (PKCS #3)
- Password-based cryptography (PKCS #5)
- Elliptic curve cryptography (PKCS #13)
- And more

RSA, part of EMC corp.

RSA, the algorithm

IETF

- Protocol standards
  - Transport Layer Security (RFC 5246)
  - Kerberos (RFC 4120)
  - Secure Shell Transport Layer Protocol (RFC 4253)
  - Extensions to protocols
- Crypto algorithm standards
  - Camellia (RFC 3713)
- Also a good source of April fools jokes

Selection by competition

- Trend towards competition-based standards selection
  - Selection committee puts out a call for submissions
  - All submissions meeting requirements are available for public scrutiny
  - Submitter's attack each other's proposals
- Benefits
  - Access to best cryptographers and cryptanalysts in the world
  - Free labor for selection committee
  - Papers and dissertations for academics
  - Fame and glory
    - With a very select crowd

NIST competition-based projects

- Advanced Encryption Standard
  - FIPS standard published 11/26/2001
  - Rijndael
    - Vincent Rijmen and Joan Daemen
- SHA-3
  - Competition winner announced 10/2/2012
  - Keccak
    - Guido Bertoni, Joan Daemen, Michaël Peeters and Gilles Van Assche.

Other projects

- NESSIE (New European Schemes for Signatures, Integrity and Encryption)
  - Selected symmetric and asymmetric primitives
  - All stream cipher submissions were defeated
- eSTREAM
  - Stream cipher competition organized by ECRYPT
  - Inspired by results of NESSIE
- CRYPTREC (Cryptography Research and Evaluation Committees)
  - Japanese Government

Closing Remarks

Summary

- Cryptology is the study of codes
  - Cryptography is the study of designing codes
  - Cryptanalysis is the study of breaking codes
- The only thing that should be considered secret is the key
- Encryption algorithms generally fall into two categories
  - Symmetric
  - Asymmetric
- Ciphers provide confidentiality
- Hash functions provide integrity
- MACs and digital signatures provide integrity and authentication

Summary (continued)

- There are several types of attacks possible
  - Each have different restrictions and relative difficulty
- An algorithm is broken when it no longer provides the security properties that it should
- “Encrypted” is not synonymous with “secure”
- There are several bodies that dictate standards

Key lessons

- "the whole is greater than the sum of the parts"
- You cannot just look for the best crypto
  - You need the right crypto for the right application, used in the right way
- The entire system has properties
  - Some properties of the system may negate the properties provided by the crypto
- There are many factors that come into play when choosing a cryptographic algorithm or protocol

More resources

- More suggested reading
  - Handbook of Applied Cryptography
    - An excellent reference for state of the art in 1997
    - Available in text or digital format (http://www.cacr.math.uwaterloo.ca/hac/)

More resources

- NIST cryptographic technology group
  - http://csrc.nist.gov/groups/ST/
- NIST cryptographic toolkit
  - http://csrc.nist.gov/groups/ST/toolkit/index.html
  - Publications of standards and recommendations
- IACR eprint archive
  - New papers in cryptology (not peer reviewed)
  - http://eprint.iacr.org/
- eSTREAM project
  - http://www.ecrypt.eu.org/stream/