EXAM 2 STUDY GUIDE

Octal is a base-8 number system, using digits 0 - 7.

❖ Three binary digits represent one octal digit.

0 = 000	1 = 001	2 = 010	3 = 011	4 = 100
5 = 100	6 = 110	7 = 111		

So 75 in octal — 7 = 111 and 5 = 101, so 75 is 111 101 in binary.

❖ Use power series expansions to convert from octal to decimal.

75 in octal — **7***8^1 + **5***8^0 = 61 in decimal

Hexadecimal is the base-16 number system, using digits 0 - 9 and letters A - F, with A representing 10 and F representing 15.

❖ Four binary digits represent one hexadecimal digit.

1 = 0001	2 = 0010	3 = 0011	4 = 0100	5 = 0101
6 = 0110	7 = 0111	8= 1000	9 = 1001	A = 1010
B = 1011	C = 1100	D = 1101	E = 1110	F = 1111

So A5 in hexadecimal — A = 1010 and 5 is 0101, so 1010 0101 in binary.

❖ To convert from binary to hex, take 4 bits at a time from right to left, replacing with hex digits; add 0's to the left if the last section isn't an even 4 bits.

1011111010101 — 0001 | 0111 | 1101 | 0101 = 17D5

❖ Use power series expansions to convert from hexadecimal to decimal.

A5 in hexadecimal — **A***16^1 (10*16^1) + **5***16^0 = 165 decimal

In base-2 numbering systems (**binary**), with *x* bits, you can only represent 2^x numbers. Since only a fixed amount of positive and numbers can be represented, you might get wrong results in arithmetic operations if they **overflow**.

❖ Overflow has only occurred in two's complement if you add two positive numbers and get a negative result, or if you add two negative numbers and get a positive result.

❖ Negating two's complement — inverse all the bits, then add 1.

❖ To subtract in two's complement, negate the subtracted number, then add the two.

4-bit one's complement number system

0111	0110	0101	0100	0011	0010	0001	0000	1111	1110	1101	1100	1011	1010	1001	1000
+7	+6	+5	+4	+3	+2	+1	+0	-0	-1	-2	-3	-4	-5	-6	-7

❖ In **one's complement**, positive numbers are the same, but negative numbers are the bit complement of the corresponding positive value, and always start with 1.

-7 in one's complement — 7 is 0111, the bit complement is 1000, so -7 is 1000.

ADVANCED ENCRYPTION STANDARD

AES, or Advanced Encryption Standard, uses a 128-bit, 192-bit, or 256-bit key, and encrypts in 128-bit blocks at a time; it is a symmetric algorithm, just like DES.

128 bits = 16 bytes of 8 bits each – interpreted in "column major order"

$(a_{0,0}\,|a_{1,0}\,|a_{2,0}\,|a_{3,0}\,a_{0,1}\,|a_{1,1}\,|a_{2,1}\,|a_{3,1}\,|\,a_{0,2}\,|a_{1,2}\,|a_{2,2}\,|a_{3,2}\,|\,a_{0,3}\,|a_{1,3}\,|a_{2,3}\,|a_{3,3})$

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$

- ❖ Unliked DES, it's based on the Rijndael algorithm, and has 10 rounds; each round transforms a 128-bit array, arranged in a 4-byte by 4-byte square array called a **state** (shown right).
- ❖ The **initial state** is the plaintext XOR'd with the key, and each state uses the previous state to produce the next state; the cipher text is the result of the final round.
- ❖ For each round, there's a **SubBytes** step (byte substitution, with 1 S-box used on every byte), a **ShiftRows** step (permutation), **MixColumns** step (essentially a Hill cipher on each column), and an **AddRoundkey** step, where the state is XOR'd with the round key.
- ❖ In the ShiftRows step, the first row is unchanged, the 2nd row does 1 circular byte shift left, third row does 2 shifts, fourth row does 3.

BLOCK CIPHER MODES OF OPERATION

A **mode of operation** describes the particular way a block cipher encrypts a sequence of message blocks.

- ❖ **Padding** a block is necessary if the last block is shorter than the required block size; padding must be unambiguous, and padding can be done with a non-data value, such as null, or with the counts of the pad size.

Electronic codebook is the simplest mode of operation, where the message is broken into blocks and each block is independently encrypted.

- ❖ Allows for parallel encryption, and can tolerate loss or damage of a block.
- ❖ Patterns in plaintext are repeated in cipher, so no diffusion; best for short messages.

Cipher block chaining is a mode of operation where messages are broken into blocks, and the blocks are linked in the encryption operation.

- ❖ Each plaintext block is XOR'd with the previous encrypted block (started off with an initial vector), then encrypted.
- ❖ It depends on all blocks before it, so damage or loss to one block affects all blocks.
- ❖ Needs an initialization vector, which is essentially another secret key that must be shared by both parties, and is a vulnerability.
- ❖ Mainly used for bulk data encryption and authentication.

Cipher feedback is a mode of operation where the cipher text from one block is re-encryped and XOR'd to encrypt the next plaintext block.

- ❖ A very common streaming mode, because it cannot be done in parallel and there's a stall while each block is encrypted; mainly used for stream data encryption and authentication.
- ❖ Errors propagate for several blocks after the error.

Output feedback is a mode of operation that acts like a one-time pad and XORs it with the plaintext.

- ❖ The initial vector is needed, and each key is an encryption of the previous key; the cipher text is then created by XORing the plaintext with the current key.
- ❖ Bit errors don't propagate, but the same key can never be reused, and the sender and receiver must be in sync; main uses for stream encryption on noisy channels.

Counter is a mode of operation very similar to output feedback but the pads are calculated differently.

- ❖ The first pad is an encryption of a seed value, the second pad is an encryption of the seed plus 1, third pad is encryption of seed plus 2, etc.
- ❖ Allows for parallel encryptions, and for random access to encrypted data blocks, but like OFB, the same pad can never be reused; mainly used for high-speed network encryptions.

INTEGRITY CHECKING

A **hash function** is an algorithm that takes any size data as input and produces a fixed-size bit string as output, called the **hash value**, **hash code**, or **message digest**.

- ❖ A **cryptographic hash function** is a type of hash function that should be one-way (easy to compute the hash from the message, but not the message from the hash), and collision resistant.
- ❖ A **birthday attack** proves that it is possible to do a brute-force attack to a cryptographic hash function if Eve tries enough times to create a collision; a b-bit hash is produced, the possible number of hash values is 2^b, if Eve generates enough messages, it'll only take $2^{b/2}$ messages to find 2 that collide
- ❖ Hashes are useful for digital signatures, integrity checks, and **Message Authentication Codes** (MAC), where a sender computes the hash of the shared secret key and the message.
- ❖ MACs are computed by using the cryptographic hash function on a secret key, followed by the message; Alice then sends the original message with the MAC; if Bob can compute the same MAC with the secret key and original message, Bob knows nothing's up.

A **checksum** is a small value that's computed using the bits from the message, and is used to check for errors or changes in files, as a change in one bit will often change the entire value.

- ❖ Computing a checksum using the **modular sum** algorithm:
  - For 4-bit checksums, divide input into 4-bit chunks
  - Add all the chunks to each other as unsigned binary numbers, discarding overflow bits
  - Use the two's complement of the result as the checksum value
  - Append the checksum to the message
- ❖ Validating a checksum:
  - Divide message into same size chunks, including checksum
  - If the result is not all 0's, then something's up
- ❖ Single-bit errors will be detected, but 2-bit errors are much less likely to be detected the larger the chunks are (1/n, where n is the number of bits in a chunk).

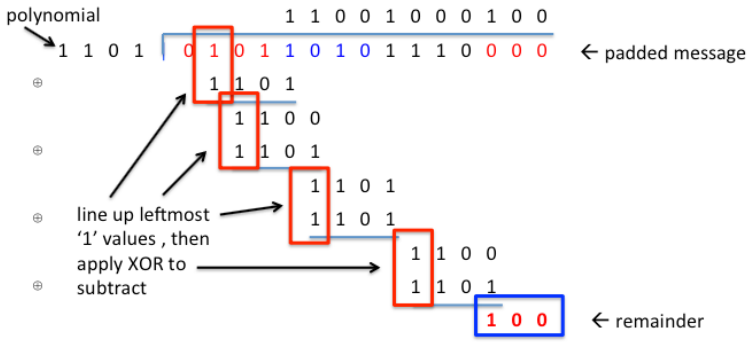
A **cyclic redundancy check** is a checksum made by calculating the remainder of the message divided by a polynomial.

- ❖ A CRC is defined by the power of the polynomial (CRC-3 for a max), the remainder is always one bit smaller then the power, but the final checksum is always one bit larger than the power (4 bits for CRC-3).
- ❖ Computing a CRC with the message 5AE and polynomial $x^3 + x^2 + 1$:

Convert polynomial into binary - 1101, corresponding to coefficient of each variable, $1x^3 + 1x^2 + 0x^1 + 1$

Convert message into binary - 0101 1010 1110, then pad with 3 0's (for CRC-3).

Line up leftmost '1' values and apply XOR to subtract



Pad remainder as necessary to get required number of bits

- ❖ Validating a CRC:
  - Divide the appended message (where the remainder is *not* padded with extra bits) by the polynomial

The remainder, also called the **syndrome** (different from the CRC, which is the first remainder), should be all 0's, which means there were no changes to the original message.

A **digital signature** provides authentication and non-repudiation on messages, as well as integrity by encrypting the message with Alice's private key, and Bob uses Alice's public key. When Bob uses Alice's public key to decrypt, he should receive the same message twice, ensuring Alice really sent it.

❖ In practice, digital signatures are not used on entire messages; usually, this is just done on the cryptographic hashes of messages.

PUBLIC KEY RSA

Euler's totient function, **$\phi(n)$**, is defined as the number of positive integers less than or equal to n that are relatively prime to n.

❖ i.e. $\phi(1) = 1$, $\phi(2) = 1$, $\phi(3) = 2$, $\phi(7) = 6$

❖ If n is prime, then $\phi(n) = n - 1$.

❖ If p and q are prime or relatively prime, $p \neq q$, and $n = pq$, then $\phi(n) = \phi(pq)$, which equals $\phi(p) \cdot \phi(q)$.

❖ **Euler's theorem** — $a^{\phi(n)} \bmod n = 1$, where a and n are relatively prime.

The **RSA encryption algorithm** is a public key system.

❖ Step-by-step encryption and decryption:

Alice encrypts a message using Bob's public key, $PU_{Bob} = \{e, n\}$ by computing

$C = M^e \bmod n$, where the message length is less than n

Bob decrypts the cipher text by using his private key, $PR_{Bob} = \{d, n\}$ by computing

$M = C^d \bmod n$

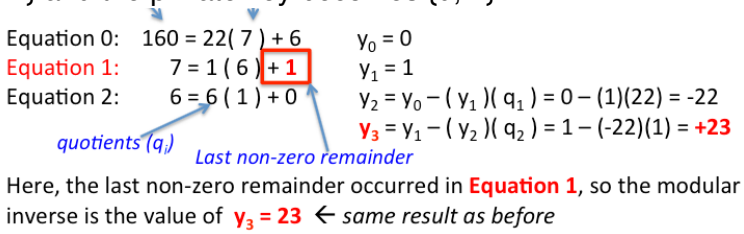
Where e and d are inverses of each other mod $\phi(n)$. Essentially, what happens is $C^d \bmod n = M^{e \cdot d} \bmod n$, $M^{1+k[\phi(n)]} \bmod n$, then $[(M^1 \bmod n)(M^{\phi(n)} \bmod n)^k] \bmod n$, which eventually comes out to M.

❖ To create the key for RSA, after picking p and q and computing n, an e is randomly selected where the GCD of e and $\phi(n)$ is 0; d is selected to be the modular inverse of e mod $\phi(n)$; the public key becomes {e, n} and the private key becomes {d, n}.

❖ One sure-fire way of finding d is by using the **Extended Euclidean Algorithm**:

Given $\phi(n) = 160$, and $e = 7$:

Always start with $y_0 = 0$ and $y_1 = 1$, and after that, $y_i = y_{i-2} - (y_{i-1})(q_{i-2})$.



Here, the last non-zero remainder occurred in **Equation 1**, so the modular inverse is the value of $y_3 = 23$ ← *same result as before*

The desired result is the y value whose index is **two more than the index for the last non-zero remainder**; here, y's index should be 3.

DIFFIE-HELLMAN KEY EXCHANGE

A number a is a **primitive root** of prime number p if and only if $a^1 \bmod p$, $a^2 \bmod p$, … $a^{p-1} \bmod p$ are all distinct.

❖ i.e. 3 and 5 are primitive roots of 7; 3 mod 7 = 3, 9 mod 7 = 2, etc.

The **Diffie-Hellman Key Exchange** allows two parties to compute a shared secret key without disclosing private keys to each other.

- ❖ Given x, g, and p, find k so that $x = g^k \bmod p$, where p is a prime number and a is a primitive root of q.
- ❖ Step-by-step:

  Alice picks a random $x < q$, and computes $X = a^x \bmod q$, and sends this to Bob.
  Bob picks a random $y < q$ and computes $Y = a^y \bmod q$, and sends this to Alice.
  Alice computes $K = Y^x \bmod q$, and Bob computes $K = X^y \bmod q$, which leads to the same key.

$Y^x \bmod q = a^{yx} \bmod q = a^{xy} \bmod q = X^y \bmod q$.

- ❖ Typically, there is a hierarchy of keys, where there is a **session key**, temporarily used to encrypt data between users, and a **master key**, used to encrypt session keys and shared by the user and distribution center.

OPERATING SYSTEMS

The **operating system** is the collection of software that provides the framework for application programs to execute on the computer by managing hardware resources.

- ❖ The **kernel** is the set of essential components for managing processor(s), memory and I/O devices that often operates with higher privileges than the user.
- ❖ Background OS processes that run sans user intervention are called **daemons** or **services**.
- ❖ I/O devices are controlled using **device drivers**, which provide an **API** (application programming interface) for software interaction.
- ❖ Every program is an instance of a **process**, which is loaded into RAM if used, in persistent storage if not; every process has its own unique ID, which allows multiple instances of the same program running simultaneously using **time slicing**, which means each process gets its own share of the CPU.
- ❖ Processes can communicate with each other via pipes/sockets, file systems (by reading and writing to shared files in persistent storage), by using shared RAM, using signals and listeners, or invoking subroutines in other processes.

File and folder **permissions** for reading, writing, and executing are maintained for user, group, and world.

- ❖ By default, the owner has full permissions while permissions are denied to all other users.
- ❖ Sometimes, permissions are abbreviated as octal numbers where 1 is read, 2 is write, and 4 is execute, so the number 7 would mean that user/group has all privileges.

In order to run, a program must have memory allocated to it, also called the **address spac**e for the program.

NETWORKING CONCEPTS

In layered network models, higher layers use the services of lower layers via **encapsulation**; each layer can be implemented in either hardware or software, but the bottom layer must be hardware.

- ❖ The internet uses the layered network called the **Internet Protocol Suite**, which consists of the physical layer (transmits bits via connected nodes), the link layer (LAN, moves frames between directly connected hosts), network layer (Internet-wide packet communication), transport (application-level communications), and the application layer.