Chapters 2, 20, 21 are going to be on the exam.
Most likely next thursday.
punctuation in the ciphertext passages and we should just get rid of it

- Hashing: like a remapping, symbolic representation of something else. We shouldn't be able to undo it and achieve the original result
    - we can rehash the original piece and get the same hash to confirm
    - one way hashing, can do it but cannot be undone
    - Used with encryption, does not replace encryption
    - We can use hashed values to identify encrypted messages, basically providing a signature
- Mode of operation (ECB and COUNTER) is some type of additional step that allows us to further encrypt
    - we want to stop patterns
        - if we had 64 bit block cipher and we had 65 bits… we would need to redo the block again for the last bit and that allows for identifying patterns
        - if we provide a mode of operation this adds an additional layer so that it becomes more difficult
- Hashing requirements:
    - can be applied to block of data any size
        - bit size would not change if we used a 512 hash for a single character or a million characters
        - since we can redo the process we can determine what the key is by rehashing a possible candidate
    - produces a fixed-length output
    - H(x) is relatively easy to compute for any given x
    - one way or preimage resistant
        - computationally infeasible to find x such that H(x)=h
        - can't determine what the original message was
    - Computationally infeasible to find y != x such that H(y) = H(x)
        - two keys do not hash to the same value
    - collision resistant or strong collision resistance
        - Computationally infeasible to find any pair (x, y) such that H(x) = H(y)
    - Simple one way function

	Bit 1	Bit 2	• • •	Bit n
Block 1	b_{11}	b_{21}		b_{n1}
Block 2	b_{12}	b_{22}		b_{n2}
	• • •	• • •	• • •	• • •
Block m	b_{1m}	b_{2m}		b_{nm}
Hash code	C_1	C_2		C_n

Figure 21.1 Simple Hash Function Using Bitwise XOR

- Going downward allows us to change up the XOR enough

- SHA (Secure Hash Algorithm):
  - Originally developed by NIST and published in 93
  - Quickly revised in 95 as SHA-1 to introduce stronger hashing values (160-bit)
  - Even further revised in 2002:
    - Adds 3 additional versions of SHA
    - SHA-256, SHA-384, SHA-512
    - with 256/384/512-bit hash values
    - Same basic structure as SHA-1 but greater security
  - Older Versions phased out by 2010 but are still in use today

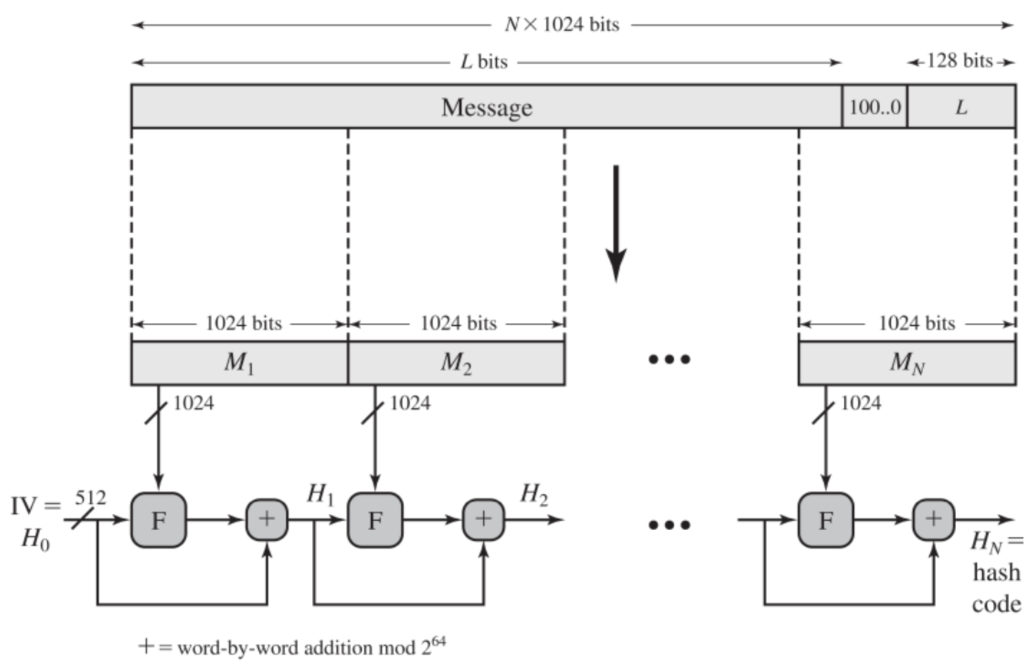

Figure 21.2 Message Digest Generation Using SHA-512

- Hash-Based Message Authentication Code (HMAC):
    - Interest in developing a MAC derived from a cryptographic hash code
        - Cryptographic has functions generally execute faster
        - library code is widely available
        - SHA-1 was not designed for use as a MAC (continue here)
    - Security depends on the cryptographic strength of the underlying hash function
    - for a given level of effort on messages generated by a legitimate user and seen by the attacker, the probability of successful attack on HMAC is equivalent to one of the following attacks on the embedded has function:
        - Either attacker computes output even with random secret IV
            - brute force key O(2^n) or use birthday attack
        - Or attacker finds collisions in hash function even when IV is random and secret
            - ie. find M and M' such that H(M) = H(M')
            - birthday attack O(2^n/2)
            - MD5 secure in HMAC since only observe
- Public-Key Encryption:
    - Introduced by Whitfield Diffie and Martin Hellman in 76
    - Based on purely mathematical functions
        - lets two people create a unique token and if we both know the secret we can create the same value
        - uses a private key and a public key, by combining them the receiver can do the same steps with the same keys and will get the same
        - you need both key sets for this to work
    - Distinguished property: Asymmetry
        - uses two separate keys: public and private key pair
        - public key is made public for anyone to see

Public Key Encryption:

algorithm	Digital signature	symm key	key encrypt
RSA:	Yes	Yes	yes
Diffie hellman:	No	Yes	No
DSS:	Yes	No	No
Elliptic Curve:	Yes	Yes	Yes