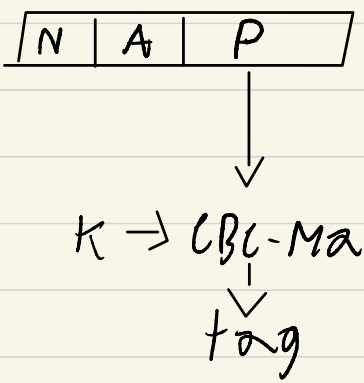
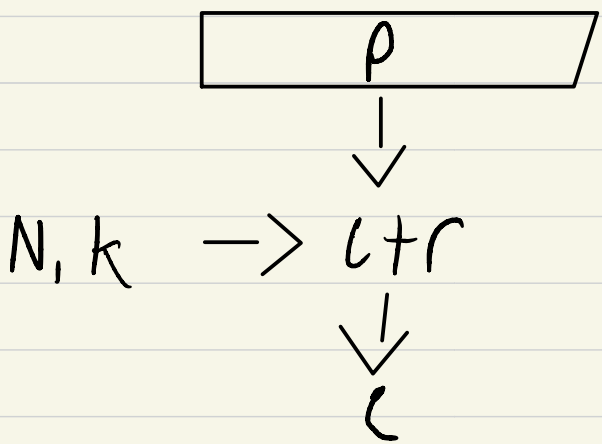
CSC 152 3/10 Notes

CCM Mode (AE) [Authenticated Encryption]

N = nounce (doesn't match any ctr blk)
A = Authentication (not encrypted)
P = Plain text (encrypted)





Send: N, tag, A, C

- uses only one key
- faster than 1 at a time

Init: k, Pt, Ptlen, C

C || tag || N
Ptlen 16 12

decrypt: k, ct, ctlen, pt

ctlen − 28

return 1: if Authentic
0: if not

Week 8 Module

Authentication with universal Hashing

- Message authentication
  - † Verifying message is not altered
  - † normally done with TagGen
    - Prob of manipulation must be less than $\frac{1}{2^n}$

- Wegman-Carter Authentication
  - † n is a positive int
  - † TagGen $(h,f),M_j) = (h(M_j) + f(j)) \mod n$

Video: Definition

- 5 to 10 clock cycles to run cryptographic hash
- Universal Hash Function

Let H be a collection of hash functions $A \to B$.

domain, co-Domain * ϵ = epsilon *

H is ϵ-almost-universal if prob of winning is $\leq \epsilon$

1) Adversary chooses $a \neq b \in A$
2) $h \in H$ is chosen randomly

Adversary wins if $h(a) = h(b)$

Ex)

	h_0	h_1	h_2	h_3	$Z_6 \to Z_4$
0	0	1	1	0	
1	1	3	1	2	ϵ-AU for
2	2	0	2	2	$\epsilon = 1/2$
3	1	3	3	3	$1/2$-AU
4	2	2	3	2	
5	0	0	0	1	

domain; co-Domain

(a,b)

$(0,1) = 1/4$ $(0,2) = 0$ $(0,3) = 0$

$(0,4) = 0$ $(0,5) = 1/4$ $(2,4) = 1/2$

Polynomial Hash

Let p be prime

Let $(x_1, x_2, \ldots, x_n)$ each be in Z_p

Let $k \in Z_p$

hash result $= x_1 k^n + x_2 k^{n-1} + \cdots + x_n k^1 \mod p$

Let $(x_1, x_2, \ldots, x_n) \neq (x_1', x_2', \ldots, x_n')$

mod p implied

$\Pr[x_1 k^n + x_2 k^{n-1} + \cdots + x_n k = x_1 k^n + x_2 k^{n-1} + \cdots + x_n k]$

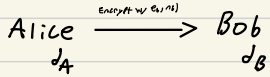
$\Pr[(x_1 - x_1')k^n + (x_2 - x_2')k^{n-1} + \cdots + (x_n - x_n')k = 0] \leq \frac{n}{p}$ good if n is small, p is large

- at least 1 nonzero coefficient
- results is upto degree n polynomial
- upto n values of k cause result is 0

PolyHash is $\frac{n}{p}$ almost universal when there are no more than n data elements being hashed and prime MOD p is used.

3/24 Notes

Asymetric cryptography



Public
(e_A, n_A)
(e_B, n_B)

RSA problem (Hard problem)

Given: e & n, $x^e \bmod n$
Find: x

Eulers theorem: if $a \in \mathbb{Z}_n^*$ then $a^{|\mathbb{Z}_n^*|} \bmod n = 1$

$*\Phi(n) = |\mathbb{Z}_n^*|*$

RSA key Generation (textbook RSA)

Math topic

$\mathbb{Z}_n = \{0, 1, \ldots n\}$
$\mathbb{Z}_n^* = \{x \in \mathbb{Z}_n \mid gcd(x, n) = 1\}$
(x has a multiplicative inverse only if $gcd(x,n)=1$)

1) Choose random prime p and q
Set $n = pq$ and $\phi(n) = (p-1)(q-1)$

2) Choose $e > 1$ so that $gcd(e, \phi(n)) = 1$

3) Set $d = e^{-1} \bmod \phi(n)$

Publish public key (e, n)
and keep private d

Example

Key Generation
$p = 31$ $n = 403$
$q = 13$ $\phi(n) = (p-1)(q-1) = 30 \times 12 = 360$

e: $gcd(2, 360) > 1$ ✗
$gcd(3, 360) > 1$ ✗
$gcd(5, 360) > 1$ ✗
$gcd(7, 360) = 1$ ✓

$e = 7$

$d = 7^{-1} \bmod 360$
$= 103$

Public key $= (7, 403)$
Private key $= 103$

test $(x = 400)$

Encrypt
$y = 400^7 \bmod 403$
$= 231$

Decrypt
$x = 231^{103} \bmod 403$
$= 400$

*y = cypher text
x = plain text*

Encryption of x

$y = x^e \bmod n$

Decryption of y

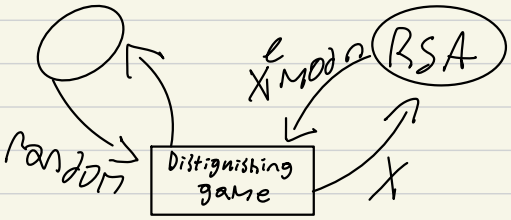
$x = y^d \bmod n$

large numbers algorithm

$O(n)$ is not fast enough

$O(\log n)$ is fast enough

Textbook RSA problems

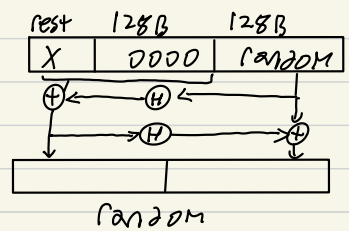


Attacks

1) Send x twice. If the same then RSA (Deterministic)
2) Send 0 if result is 0 then RSA (Math)

OAEP

Optimal Asymetric encryption padding



H = hash function

Encrypt

$X' = OAEP(x)$
$y = (x')^e \bmod n$

Decryption

$x' = y^d \bmod n$
$X = OAED(x')$
* if x has zero structure extract X *

Module 9 Lecture

Multiply

$(x^y)^2 = x^{2y} \Rightarrow x^{y\|0}$ "11" = (2+1)*

$(x^y)^2 x = x^{2y+1} \Rightarrow x^{y\|1}$

pow(x,y): $\Theta(n)$

```
let y = y1 y2 ... yn  where yi ∈ {0,1}
acc = 1
for i = 1 to n
  acc = acc * acc
  if yi == 1
    acc = acc * x
return acc
```

GCD

let d be a divisor of x and y

compute x mod y

$x = yq + r \quad 0 \le r < y$

(x ↑ mult of d; yq ↑ mult of d; r ↑ mult of d)

$gcd(x,y) = gcd(y, x \% y)$

$gcd(x,0) = x$

gcd(x,y) $\Theta(\log(\max(x,y)))$

```
while y != 0:
    t = y
    y = x % y
    x = t
return x
```

$gcd(x,y) \to (a,b)$

$\Rightarrow ax + by = gcd(x,y)$

Multiplicative inverse

$5^{-1} \bmod 31$

$gcd(31,5) \to (a,b)$

$\Rightarrow (a)(31) + (b)(5) = 1$

$= (31a + 5b) \% 31 = 1 \% 31$

$= 5b \% 31 = 1$

(5, b ↑ inverses)

if $b < 0$ add 31 to it until its positive

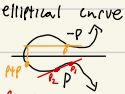
Week 10/11 Module

Groups

- real numbers don't have a hard logarithmic problem
- Fields are a pair of groups

- Group
  - † set of objects G
  - † operation $a \times a \to a$ (mult or add)
  - † identity (0 or 1)
  - † inverses: $\forall x \in a, \exists y \in a$ $x \circ y =$ identity
  - † associative and commutative

elliptical curve group additive



G: integer points on line
identity: θ inverse of $(x,y) = (x,-y)$

operation:
$P + \theta = P$
$\theta + \theta = \theta$

if a is a group and $x \in G$
then $x^{|G|} = 1$

$x^i = 1 \cdot \underbrace{x \cdot x \cdot x}_{i \text{ times}}$

if G and H are groups that use the same operation and $H \subseteq a$,
Then H is a subgroup of a

RSA Signature

Alice → Bob: data, sig

sig = sign(data):
return $data^d \bmod n$

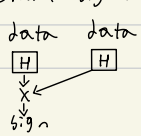
verify(data, sig)
return $sig^R \bmod n == data$

RSA problem
† given $x^R \bmod n$, e, n find x
  is hard
† implies $x^2 \bmod n$
  hard w/o d

Sign w/ Hash

sig = sign(data)
  x = H(data)
  return $x^d \bmod n$

verify(data, sig)
  x = H(data)
  return $sig^e \bmod n == x$



Example

P = (5,1)
order(P) = 19

Alice → Bob: 8(5,1)

$0 \le X \le 19$
X = 8
Bob → Alice: X = 10 (10(5,1))

key = 8(10(5,1)) = 10(8(5,1))
Alice: 8(5,1); Bob: 10(8(5,1))

8(5,1)
1000
$(0 \cdot 2 + (5,1)) \cdot 2 \cdot 2 \cdot 2$
double acc, double acc, add base

$= (5,1) \cdot 2 \cdot 2 \cdot 2$
$= (6,3) \cdot 2 \cdot 2$
$= (3,1) \cdot 2$
$= (13,7)$

10 · (13,7)
1010
$(0 \cdot 2 + (13,7)) \cdot 2 + (13,7)) \cdot 2$
$= (16,4)$ [Bob key]

8 · (10(5,1)))
$= 8(7,11)$
1000
$= (7,11) \cdot 2 \cdot 2 \cdot 2$
$= 3,1$

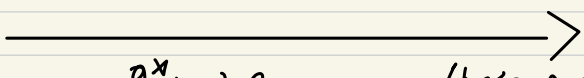
```
divisionless_mod_final(){
X = (X >> a) + (x & ((1<<a) - 1))   // Maybe x ≥ p
t = (p - 1) - x)   // t is neg if x ≥ p  pos if x < p
t = t >> 63   // t = 1^64 if x ≥ p   t = 0^64 if x < p
return x - (p & t)
```

int64_t →

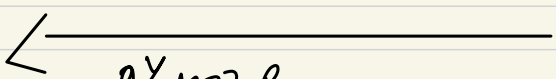
↑
p if x ≥ p else 0

Diffie-Hellman key exchange (1970's)

Alice	known	Bob
chose random	prime p	
$0 < x < p$	generator G	



$g^x \bmod p$ — chose random $0 < y < p$



$g^y \bmod p$

knows	Adversary knows	knows
x	g^x	y
g^y	g^y	g^x

$K = (g^y)^x = g^{yx} \bmod p$ $K = (g^x)^y = g^{xy} \bmod p$

secure against passive adversary

Picking Random primes

n bit prime

Theorem: there are infinite primes

Proof Sketch:
for contradiction assume there are finite primes
$p_1, p_2, p_3, \ldots p_n$
Let $q = p_1, p_2, p_3, \ldots, p_n$
Then $q+1$ is not prime because it is larger than $p_1 \ldots p_n$
Let p be a prime factor of $q+1$
Then both q and $q+1$ are a multiple of p which is impossible.
Therefore there are infinite primes.

Density of primes: Pr[random n bit # is prime] $\approx \frac{1}{n}$

idea:

```
gen-prime(n):
    DO
      p = random n bit #      } ≈ n iterations
    WHILE (not is_prime(n))
    return n
```

```
is_prime(p)   O(log p)
    seen_p-1 = false
    for i=1 to confidence level
        X = random   // 0<X<p
        temp = X^(p-1) mod p
        if temp != 1 or p-1   return false
        if temp = p-1
            seen_p-1 = true
    return true  seen_p-1
```

- Basic but works
- More thorough

Number theory

$$\Pr_{0<x<p}\left[x^{\frac{p-1}{2}} \bmod p \in \{1, p-1\} \mid p \text{ is prime}\right] = 1$$

$$\Pr_{0<x<p}\left[x^{\frac{p-1}{2}} \bmod p \in \{1, p-1\} \mid p \text{ is not prime}\right] \leq \frac{1}{2}$$

Let $0 < x_1, \ldots, x_q < p$ be random

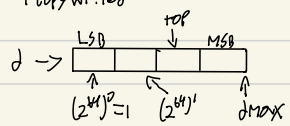
$$\Pr\left[\text{all } x_i^{\frac{p-1}{2}} \bmod p \in \{1, p-1\} \mid p \text{ is prime}\right] = 1$$

$$\Pr\left[\text{all } x_i^{\frac{p-1}{2}} \bmod p \in \{1, p-1\} \mid p \text{ is not prime}\right] \leq \left(\frac{1}{2}\right)^q$$

prob of false positive

BigNum Notes

-found on github for code + copywrited



gen_prime.c

```
#include <openssl/bn.h>

void gen_prime(BigNum *r, int nbits, int confidence level) {
    BigNum *x = BN_new();
    do {
      BN_rand(x, nbits, BN_RAND_TOP_ANY, BN_RAND_BOTTOM_ODD);
    }while(! is_prime(x));
    BN_copy(r, x);
    BN_free(x);
}

int is_prime(BigNum *p, int confidence){
    int seenPminus1 = 0;
    BigNum *pMinus1 = BN_new();
    BigNum *pMinus1over2 = BN_new();
    BigNum *x = BN_new();
    BigNum *t = BN_new();
    BN_CTX *ctx = BN_CTX_new();
    BN_sub(pMinus1, p, BN_value_one());
    BN_rshift1(pMinus1over2, pMinus1);
    for(int i=0; i< confidence; i++){
        BN_rand_range(x, pMinus1);          //range [0,p-2]
        BN_add(x, x, BN_value_one());  //range [1,p-1]
        BN_mod_exp(t, x, pMinus1over2, p, ctx);
        if((BN_cmp(t, BN_value_one()) != 0) &&
              BN_cmp(t, pMinus1) != 0))
            return 0;
        if(BN_cmp(t, p-1) == 0)
            seenPminus1 = 1
    }
    return seenPminus1;
}
```

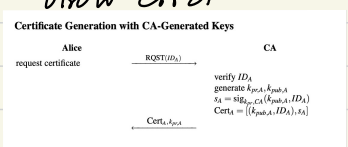
need to free all news

Week 13/14 Module

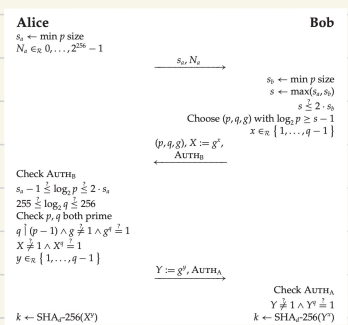
Readings

Ch 13.3

Man in the Middle attack
- possible when public keys aren't authenticated
- adding a signature with help as when attacker change key so will ID which will show error



Key negotiation



Tweakable Block Ciphers

Signature of block cipher
$E: \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$
Tweakable block cipher
$\tilde{E}: \{0,1\}^k \times \{0,1\}^t \times \{0,1\}^n \to \{0,1\}^n$

- Tweakable block cipher
  - takes 3 inputs
  - t = tweak ← public
    - similar intent as nonce
  - needs to be secure

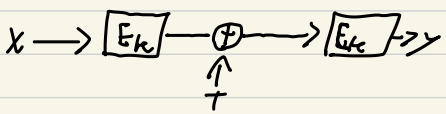
Videos

Certificates

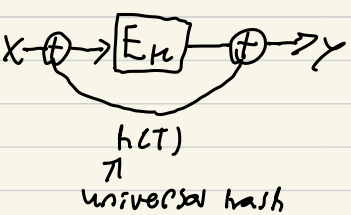
- a text file with
  - owner ID
  - Issuer
  - Valid dates
  - Valid uses
  - Owners public key
  - Issuers signature
- usage:

Alice sends: file, sig, Alices cert
Bob Does: uses Alices public key to verify sig of file

Tweak construction



$Time(\tilde{E}) = 2 \cdot time(E)$



$Time(\tilde{E}) = time(E) + Time(h)$

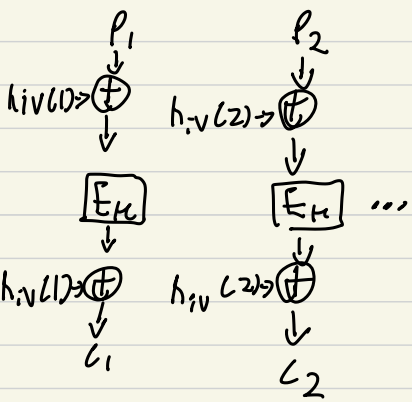
$GF(2^{128})$ mod $x^{128}+x^7+x^2+x^1+x^0$ => 0x87

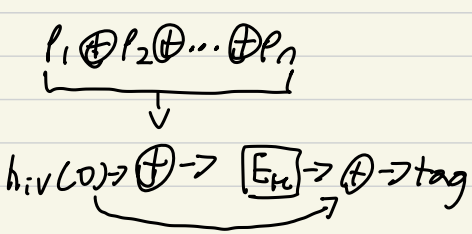
OCB Hash

$h_{iv}(0) = iv$
$h_{iv}(1) = iv \cdot 2$ ← $GF(2^{128})$
$h_{iv}(2) = iv \cdot 2 \cdot 2$
...
$h_{iv}(T) = (iv) \cdot 2^T$

```
Next_Hash(h)
hi = highbit of h
h = h<<1
if(hi==1)
  h = h ⊕ 0x87
return h
```

OCP





Time(tag) = Time(E)

Random Generation

```
fd = fopen("/dev/urandom", "r")
fread(buff, x, 1, fd)
```

Fortuna uses block cipher

Desired properties
- indistinguishable from random
- forward security: if adversary knows state they shouldn't know all future bytes
- backwards security: same but in the past

Where does it come from?

It is generated by distilling an entropy pool
- entropy: is a measurement of uncertainty
  - a process with n bits of entropy is uncertain as an n bit random string

4 sided die

1, 2, 3, 4
each $\frac{1}{4}$

2 bit string
00, 01, 10, 11
each $\frac{1}{4}$

both has 2 bit entropy

$-(\log_2 Pr)$

if X's outcome are not equally likely use a weighted avg

$$H(X) = -\sum_x Pr[X=x] \log_2 Pr[X=x]$$

distilling entropy is through cryptographic hash

$y = sha256(x)$

$H(y) = \min(H(x), 256)$

Entropy Sources

- Mouse movement
- Key presses
- Memory location of data
- Data movements
- Context switching

SSL / TLS

SSL was invented by Netscape in 1990's to enable secure web transactions

- two components
  - Handshake Protocol:
    - establishes shared master key and establish authenticity
  - Record protocol:
    - secures communication given a shared master key

OpenSSL tries to make TLS just as easy.

```
SSL_CTX  ctx = SSL_CTX_new(TLS_client_method());
SSL_CTX_load_verify_locations(ctx, "/path/to/TrustStore.pem", NULL));
BIO *bio = BIO_new_ssl_connect(ctx);
BIO_set_conn_hostname(bio, "hostname:port");
BIO_do_connect(bio);
BIO_read(bio, buf, len);
BIO_write(bio, buf, len);
```

Most TLS/HTTPS connections:

— Server sends certificate
— Client does not send certificate
— Client authenticates self via password or credit card

Client confidence comes from ClientKeyExchage and/or ServerKeyExchange which use server's certificate for authentication.

BIO abstraction allows reading and writing data of various types using a single abstraction.

```
BIO *bio = BIO_new_connect("hostname:port");
BIO_do_connect(bio);
BIO_read(bio, buf, len);
BIO_write(bio, buf, len);
```

Properties of function

let $f: Z_{10} \to Z_{10}$ be a random function

$Pr[f(0)=0]$
$Pr[f(0)=0 \mid\mid f(1)=1]$
$Pr[f(0)=0 \text{ and } f(1)=1]$

Same for permutation

Calculating Advantages

Write a distinguisher and write advantage for 1x8 and 2x4 die with 2 queries

```
Distinguisher(F)
  y1 = F(1)
  y2 = F(1)
  if y1 or y2 = 4,5,6
     Output 2x4
  else
     Output 1x8
```

$Adv = Pr[\text{guess right} \mid \text{right}] - Pr[\text{guess right} \mid \text{wrong}]$

AES

$GF(256)$ modulus $x^8+x^4+x^3+x+1$

23 x 32

00100011 x 00110010
$(x^5+x+1) \times (x^5+x^4+x)$

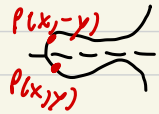
$= x^{10}+x^9+x^6+x^6+x^5+x^2 + x^5+x^4+x$
$= x^{10}+x^9+x^4+x^2+x^1$

$$x^8+x^4+x^3+x+1 \;\overline{)\; x^{10}+x^9+x^4+x^2+x \;}\; x^2+x$$
$$-\,(x^{10}+x^6+x^5+x^3+x^2)$$
$$x^9+x^6+x^5+x^4+x^3+x$$
$$-\,(x^9+x^5+x^4+x^2+x)$$
$$x^6+x^3+x^2$$

01001100
0x4C

Elliptic curves

form additive group



all point x and y are ints

$(x_3, y_3) = (x_1, y_1) + (x_2, y_2)$

$s = \frac{y_2 - y_1}{x_2 - x_1} \bmod p \quad x_1, y_1 \neq x_2, y_2$

$s = \frac{3x_1^2 + a}{2y} \bmod p \quad x_1, y_1 = x_2, y_2$

$x_3 = s^2 - x_1 - x_2 \bmod p$

$y_3 = s(x_1 - x_3) - y_1 \bmod p$

Ex)

$p = 11 \quad a = 2 \quad b = 6$

$(1,3) + (1,3)$

$s = (3 \cdot 1^2 + 2)(2 \cdot 3)^{-1} \bmod 11$

$= 5 \cdot 6^{-1} \bmod 11$

$= 5 \cdot 2 \bmod 11$

$= 10$

$x_3 = 10^2 - 1 - 1 \bmod 11$

$= 98 \bmod 11$

$= 10$

$y_3 = 10 \cdot (1 - 10) - 3 \bmod 11$ (10 mod 11)

$= 10 \cdot 2 - 3 \bmod 11$

$= 17 \bmod 11$

$= 6$

$(1,3) + (1,3) = (10, 6)$

Finding a subgroup of big group

Find subgroup size of group 1000

primes around 1000 = 1009, 1013, 1019, 1021

primes around 11 = 11, 13, 17

$p = Nq + 1$

$1013 = 92 \cdot 11 + 1$

Z_{1013}^* size 1012

$x \in Z_{1013}^* \rightarrow x^{1012} \bmod 1013 = 1$

$\rightarrow (x^{92})^{11} \bmod 1013 = 1$

x will generate a subgroup of size 1, 2, 4, 11, 22, 23, 44, 46, 92, 253, 506, 1012

if $g = x^{92} \bmod 11 \neq 1$ then subgroup size that x generates shares no factors with 92 leaves only 11

so if $g \neq 1$, g generates a size 11 subgroup

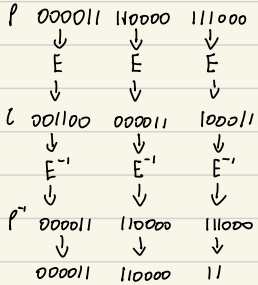
Final topics

- short C functions: bit manipulation (endian, rotate, bitset, etc), access a buffer and manipulate it, and/or implement a dataflow diagram.
- properties of random functions and random permutations
- advantage in a distinguishing game
- basic understanding of AES steps and ability to simulate them if given S-box, MixColumns matrix and GF(256) modulus.
- perform GF(2^n) operations if given modulus.
- perform encryption and decryption using ECB, CBC, CTR, OFB
- simulate sponge hash given a permutation and rate.
- know desired properties of cryptographic hash functions
- simulate polynomial hash including Horner's rule and/or divisionless mod
- determine epsilon for almost-universal hash function
- simulate RSA for key setup, encryption and signing
- compute exponents, inverses, primes using efficient algorithms learned in class.
- compute Diffie-Hellman keys using prime groups or subgroups
- find generator of a subgroup of a particular size.
- perform elliptic curve addition given the formula
- know at a high level what is in a certificate
- compute the entropy of a simple process.
- answer questions about Fortuna
- answer basic questions about the various roles that each key negotiation component plays.
- compute an OCB encryption and/or tweakable block cipher outcome similar to that seen in Homework 8.

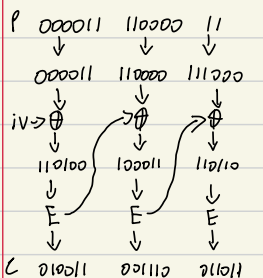
$E: \{0,1\}^b \to \{0,1\}^y$
$E(x) = ROTL(x,2)$
nonce = 101
iv = 110111
counter start at <1>
10^* padding

Encrypt 0000 1111 0000 11

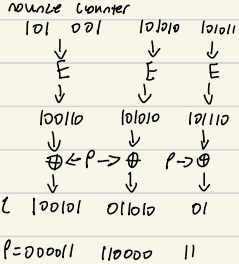
ECB



CBC



CTR



OFB