CSc466/566 Computer Security
Midterm Exam Cheat-Sheet

Christian Collberg

Mon Mar 19, 2012

A Cheat-Sheet

A.1 Modular Arithmetic

$$(a+b) \bmod n = ((a \bmod n) + (b \bmod n)) \bmod n$$
$$(a-b) \bmod n = ((a \bmod n) - (b \bmod n)) \bmod n$$
$$(a*b) \bmod n = ((a \bmod n) * (b \bmod n)) \bmod n$$
$$x^y \bmod n = \overbrace{x * x * \cdots * x}^{y} \bmod n$$

A.2 Exponents/Powers

$$x^a x^b = x^{(a+b)}$$
$$x^a y^a = (xy)^a$$
$$(x^a)^b = x^{(ab)}$$
$$x^{\left(\frac{a}{b}\right)} = \sqrt[b]{x^a}$$
$$x^{(a-b)} = \frac{x^a}{x^b}$$
$$x^{-a} = \frac{1}{x^a}$$

A.3 Logarithms

$$y = \log_b(x) \text{ iff } x = b^y$$
$$\log_b(1) = 0$$
$$\log_b(b) = 1$$
$$\log_b(xy) = \log_b(x) + \log_b(y)$$
$$\log_b\left(\frac{x}{y}\right) = \log_b(x) - \log_b(y)$$
$$\log_b(x^n) = n \log_b(x)$$
$$\log_b(x) = \log_b(c)\log_c(x) = \frac{\log_c(x)}{\log_c(b)}$$

A.4 Basic Theorems

THEOREM 1 (EULER) *Let x be any positive integer that's relatively prime to the integer $n > 0$, then $x^{\phi(n)} \bmod n = 1$.*

THEOREM 2 (COROLLARY TO EULER'S THEOREM) *Let x be any positive integer that's relatively prime to the integer $n > 0$, and let k be any positive integer, then $x^k \bmod n = x^{k \bmod \phi(n)} \bmod n$.*

THEOREM 3 (BEZOUT'S IDENTITY) *Given any integers a and b, not both zero, there exist integers i and j such that $\mathrm{GCD}(a,b) = ia + jb$.*

THEOREM 4 (COROLLARY TO EULER'S THEOREM) *Given two prime numbers p and q, integers $n = pq$ and $0 < m < n$, and an arbitrary integer k, then $m^{k\phi(n)+1} \bmod n = m^{k(p-1)(q-1)+1} \bmod n = m \bmod n$.*

THEOREM 5 (FERMAT'S LITTLE THEOREM) *Let p be a prime number and g any positive integer $g < p$, then $g^{p-1} \bmod p = 1$.*

A.5 GCD

```
func gcd(int a, int b):(int, int, int) =
    if b=0 then
        return (a,1,0)
    q ← ⌊a/b⌋
    (d,k,l) ← gcd(b, a mod b)
    return (d, l, k − lq)
```

- Use GCD to compute modular multiplicative inverses. Given $x < n$, we want to compute $y = x^{-1} \bmod n$, i.e. $yx \bmod n =$

1. The inverse of x in Z_n exists when $\text{GCD}(n, x) = 1$.

- Calculate $\text{GCD}(n, x) = (1, i, j)$ such that $1 = ix + jn$. Then $(ix + jn) \bmod n = ix \bmod n = 1$ and i is x's multiplicative inverse in Z_n.

A.6 RSA

- *Bob* (Key generation):
  1. Generate two large random primes p and q.
  2. Compute $n = pq$.
  3. Select a small odd integer e relatively prime with $\phi(n)$.
  4. Compute $\phi(n) = (p-1)(q-1)$.
  5. Compute $d = e^{-1} \bmod \phi(n)$.

  - $P_B = (e, n)$ is Bob's RSA public key.
  - $S_B = (d, n)$ is Bob' RSA private key.

- *Alice* (encrypt and send a message M to Bob):
  1. Get Bob's public key $P_B = (e, n)$.
  2. Compute $C = M^e \bmod n$.

- *Bob* (decrypt a message C received from Alice):
  1. Compute $M = C^d \bmod n$.

A.7 Diffie-Hellman Key Exchange

1. *All parties* (set-up):
   (a) Pick p, a prime number.
   (b) Pick g, a generator for Z_p.
2. *Alice*:
   (a) Pick a random $x \in Z_p, x > 0$.
   (b) Compute $X = g^x \bmod p$.
   (c) Send X to Bob.
3. *Bob*:
   (a) Pick a random $y \in Z_p, x > 0$.
   (b) Compute $Y = g^y \bmod p$.
   (c) Send Y to Alice
4. *Alice* computes the secret: $K_1 = Y^x \bmod p$.
5. *Bob* computes the secret: $K_2 = X^y \bmod p$.

A.8 Elgamal Encryption

- *Bob* (Key generation):
  1. Pick a prime p.
  2. Find a generator g for Z_p.
  3. Pick a random number x between 1 and $p-2$.
  4. Compute $y = g^x \bmod p$.

  - $P_B = (p, g, y)$ is Bob's RSA public key.
  - $S_B = x$ is Bob' RSA private key.

- *Alice* (encrypt and send a message M to Bob):
  1. Get Bob's public key $P_B = (p, g, y)$.
  2. Pick a random number k between 1 and $p-2$.
  3. Compute the ciphertext $C = (a, b)$:

$$a = g^k \bmod p$$
$$b = My^k \bmod p$$

- *Bob* (decrypt a message $C = (a, b)$ received from Alice):
  1. Compute $M = b(a^x)^{-1} \bmod p$.

A.9 RSA Signature Scheme

- *Bob* (Key generation): As before.
  - $P_B = (e, n)$ is Bob's RSA public key.
  - $S_B = (d, n)$ is Bob' RSA private key.
- *Bob* (sign a secret message M):
  1. Compute $S = M^d \bmod n$.
  2. Send M, S to Alice.
- *Alice* (verify signature S received from Bob):
  1. Receive M, S from Alice.
  2. Verify that $M \stackrel{?}{=} S^e \bmod n$.

A.10 Elgamal Signature Scheme

- *Alice* (Key generation): As before.
  1. Pick a prime p.
  2. Find a generator g for Z_p.
  3. Pick a random number x between 1 and $p-2$.

4. Compute $y = g^x \bmod p$.

- $P_A = (p, g, y)$ is Alice's RSA public key.
- $S_A = x$ is Alice' RSA private key.

- *Alice* (sign message M and send to Bob):
  1. Pick a random number k.
  2. Compute the signature $S = (a, b)$:

$$a = g^k \bmod p$$
$$b = k^{-1}(M - xa) \bmod (p-1)$$

- *Bob* (verify the signature $S = (a, b)$ received from Alice):
  1. Verify $y^a \cdot a^b \bmod p \stackrel{?}{=} g^M \bmod p$.