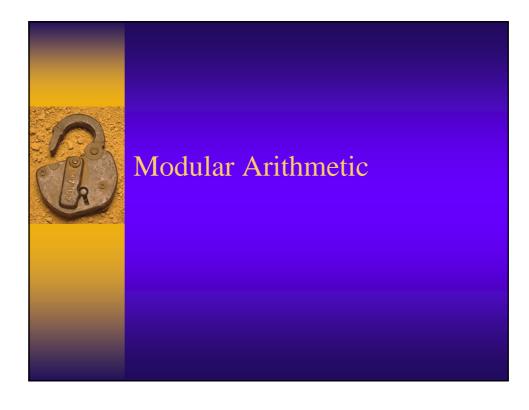


Mathematical Background for Cryptography



Overview

- ♦ Modular Arithmetic
- Relatively Prime Numbers
- Generating Prime Numbers
- ◆ Factoring





Fields

- A *field* is a set of elements with
 - two operations (+,x)
 - a "zero", s.t. ∀a, a+0=a
 - -a "one", s.t. $\forall a, a \times 1 = a$
 - $a^{-1} \text{ iff } a \times a^{-1} = 1$
 - -a iff a + (-a) = 0



Galois Fields: GF(p)

- ♦ Elements: {0,1, ...p}
- ♦ Operations: (+,x) modulo a *prime* p
- Examples:

$$4+6 \mod 7 = 3$$
 $4 \times 6 \mod 7 = 3$ $-4 = 3$ $4^{-1} = 2$

♦ Properties:

```
(a \bmod p) +/- (b \bmod p) = (a+/-b) \bmod p
(a \bmod p) \times (b \bmod p) = (a \times b) \bmod p
```



Fermat's Little Theorem

- ♦ Theorem:
 - in GF(p), \forall a≠0, a^(p-1) mod p = 1
 - note that there is a cycle here, because $a^p \mod p = a \times a^{(p-1)} \mod p = a \times 1 \mod p = a$
- ◆ Example
 - $a^6 \bmod 7 = 1 \ \forall a \neq 0 \text{ in GF}(7)$
 - hence, for any b, s.t. $b=a^2 \mod 7$, $b^3 \mod 7=1$



The Field GF(2ⁿ)

- Elements: n-bits binary vectors, e.g., (1101) in GF(24)
- Polynomial Representation
 - In GF(2^4), (1101) is represented with the polynomial $X^3 + X^2 + 1$
 - Addition = XOR of coefficients
 - Note: addition = subtraction
 - Multiplication = multiplication of polynomials modulo an irreducible polynomial
- Irreducible polynomial is divisible only by 1 and by itself
 - Analogous to a prime number
 - Usually, the polynomial X^n+X+1 is used
 - Other irreducible polynomials can also be used



The Field GF(2ⁿ) – Examples

- Addition:
 - -(0100) + (1101) = (1001)
- Multiplication
 - $(0100) \times (1101) =$ $X^{2}(X^{3}+X^{2}+1) \mod (X^{3}+X+1) =$ $X^{5}+X^{4}+X^{2} \mod (X^{3}+X+1) =$ $X^{2}(X^{3}+X+1) X^{3} X^{2} + X^{4} + X^{2} \mod (X^{3}+X+1) =$ $X^{4}+X^{3} \mod (X^{3}+X+1) =$ $X(X^{3}+X+1) X^{2} X + X^{3} = (1110)$



Relatively Prime Numbers



Relatively Prime Numbers

- Two numbers a and b are *relatively prime* if they share no common factors
 - i.e. GCD (a,b) =1, GCD = Greatest Common Divisor
- ◆ Examples
 - -GCD(21,7) = 7
 - -GCD(21,8) = 1
 - -GCD(a,p) = 1, unless a|p



Computing GCD

- Euclidean GCD Algorithm:
 - Iteratively take modulo of each other until 0

135	40
15	40
15	10
5	10
	0

105	41	
23	41	
23	18	
5	18	
5	3	
2	3	
2	1	
0 (1	



Euler Totient Function

- ◆ Definition
 - $\phi(n)$ is the number of elements a<n s.t., a is relatively prime to n (i.e. GCD(a,n)=1)
- ♦ Examples:
 - $\phi(12)=4 \{1,5,7,11\}$
 - $\phi(p)=p-1$, for a prime p {1,2,...p-1}
- Euler's Generalization of Fermat's Little Theorem
 - If a is relatively prime to n, then $a^{\phi(n)} \mod n = 1$
- Corollary: $a^b \mod n = a^{(b \mod \phi(n))} \mod n$
- Easy to compute powers
 - $e.g., a^{703} \mod 12 = a^3 \mod 12$



Calculating Inverses

- In general, in GF(n), there is not always an inverse.
- In GF(n), a has an inverse iff a is relatively prime to n
- In particular, if n is prime then there is an inverse (a<n)
- The Extended Euclidean Algorithm
 - If r is the GCD(a,b), then r=xa+yb (linear combination)
 - x and y can be computed by reversing the Euclidean Algorithm
- If a and n are relatively prime, then 1=xa+yn
 - Under mod n, we have 1=xa+0, or $x=a^{-1}$



Generating Prime Numbers



Primality Tests: Fermat

- According to Fermat's Little Theorem
 - If p is prime then $a^{(p-1)} \mod p = 1$
 - Test 1: Generate a number a<n, and test if holds
 - Test 2: Test for 2, $2^p \mod p = 2$
- Most non-primes will fail the test, but the test does not guarantee primality
 - Pseudoprimes satisfy the Fermat's condition for some a's, but are not primes
 - Carmichael numbers are non-primes for which the Fermat condition is satisfied for all a<p
 - Examples: 561, 1105
- Unfortunately, there are infinitely many Carmichael numbers
- Fortunately, they are sparse and easy to detect



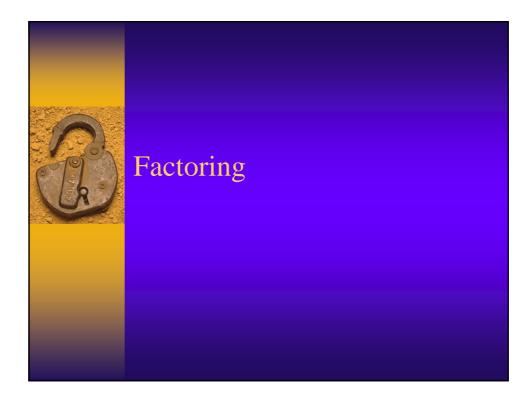
Primality Tests: Rabin-Miller

- Let $p=1+2^b$ m
 - p is odd; m is the odd number past the trailing zero bits
- ◆ Calculate z=a^m mod p
 - if $z = 1 \mod p$, then p may be a prime
- Calculate $z=a^{2^{j} m} \mod p$, for each $0 \le j \le b$
 - if $z = -1 \mod p$, then p may be a prime (-1 = p-1)
- Theorem: chances of p qualifying for an arbitrary a<1/4
- Algorithm: Repeat the test enough times to reduce the chance of coincidence



Practical Implementation

- Generate an odd number p with enough bits
 - Simply set the high and low bits to 1
- Check to see that p is not divisible by small primes
 - Usually, check against all primes <2000
- Use the Rabin-Miller test on a few a's
- Note: Primes are actually quite dense within the natural numbers (about 1:k among k-bit numbers)





Factoring

- Factoring a number means to find its prime factors
 - e.g., 60=2*2*3*5
- In general, this can be a hard problem, especially if the number has few factors, and these factors are large
 - e.g., 2¹¹³-1=3391*23279*65993*1868569*1066818132868207
- Factoring is an old problem, and is not difficult, but it can be time consuming
 - The Number Field Sieve (NFS) algorithm is considered the fastest for very large numbers
 - Has exponential run-time !!



How Hard Is It

- Number of decimal digits factored using Quadratic Number Sieve algorithm
 - 1983 71 digits
 - 1989 100 digits
 - 1993 129 digits
- QS is based on an observation of Fermat
 - Every odd composite number can be written as difference of two squares: $X^2 Y^2$ (hence (X+Y),(X-Y) are factors)
- NFS is faster than QS, and is getting better with new optimizations
 - Uses non-integers also (roots)
 - RSA-211 = $(10^{211}-1)/9$ factored in 1999
 - RSA-155 (512 bits) is more difficult and was also factored in 1999
 - 35 computing years on Unix/Pentium machines, over the course of 7 calendar months



Strong Primes

- In many cryptographic algorithms, the key is made of a product of two primes p,q
- It is desirable that p,q be hard to discover (strong primes):
 - GCD(p-1,q-1) is small
 - Both p-1 and q-1 have large prime factors p', q'
 - Also p'-1 and q'-1 have large prime factors
 - (p-1)/2 and (q-1)/2 are both prime
- This is not a formal definition, only a wishlist